

Clustering With K-Means in Python

A very common task in data analysis is that of grouping a set of objects into subsets such that all elements within a group are more similar among them than they are to the others. The practical applications of such a procedure are many: given a medical image of a group of cells, a clustering algorithm could aid in [identifying the centers of the cells](http://www.academia.edu/2496077/Automatic_Segmentation_of_Skin_Cancer_Images_using_Adaptive_Color_Clustering) (http://www.academia.edu/2496077/Automatic_Segmentation_of_Skin_Cancer_Images_using_Adaptive_Color_Clustering); looking at the GPS data of a user's mobile device, their more frequently visited locations within a certain radius can be revealed; for any set of [unlabeled observations](http://en.wikipedia.org/wiki/Unsupervised_learning) (http://en.wikipedia.org/wiki/Unsupervised_learning), clustering helps establish the existence of some sort of structure that might indicate that the data is separable.

Mathematical background

The k-means algorithm takes a dataset X of N points as input, together with a parameter K specifying how many clusters to create. The output is a set of K cluster centroids and a labeling of X that assigns each of the points in X to a unique cluster. All points within a cluster are closer in distance to their centroid than they are to any other centroid.

The mathematical condition for the K clusters and the K centroids can be expressed as:

$$\sum_{k=1}^K \sum_{x_n \in C_k} \|x_n - \mu_k\|^2$$

Minimize with respect to .

Lloyd's algorithm

Finding the solution is unfortunately [NP hard](http://en.wikipedia.org/wiki/NP-hard) (<http://en.wikipedia.org/wiki/NP-hard>). Nevertheless, an iterative method known as Lloyd's algorithm exists that converges (albeit to a local minimum) in few steps. The procedure alternates

between two operations. (1) Once a set of centroids is available, the clusters are updated to contain the points closest in distance to each centroid. (2) Given a set of clusters, the centroids are recalculated as the means of all points belonging to a cluster.

$$C_k = \{x_n : \|x_n - \mu_k\| \leq \text{all } \|x_n - \mu_l\|\} \quad (1) \quad \mu_k = \frac{1}{|C_k|} \sum_{x_n \in C_k} x_n \quad (2)$$

The two-step procedure continues until the assignments of clusters and centroids no longer change. As already mentioned, the convergence is guaranteed but

the solution might be a local minimum. In practice, the algorithm is run multiple times and averaged. For the starting set of centroids, several methods can be employed, for instance random assignment.

Below is a simple implementation of Lloyd's algorithm for performing k-means clustering in python:

```
1 import numpy as np
2
3 def cluster_points(X, mu):
4     clusters = {}
5     for x in X:
6         bestmukey = min([(i[0], np.linalg.norm(x-mu[i[0]])) \
7                           for i in enumerate(mu)], key=lambda t:t[1])[0]
8         try:
9             clusters[bestmukey].append(x)
10        except KeyError:
11            clusters[bestmukey] = [x]
12    return clusters
13
14 def reevaluate_centers(mu, clusters):
15     newmu = []
16     keys = sorted(clusters.keys())
17     for k in keys:
18         newmu.append(np.mean(clusters[k], axis = 0))
19    return newmu
20
21 def has_converged(mu, oldmu):
```

```

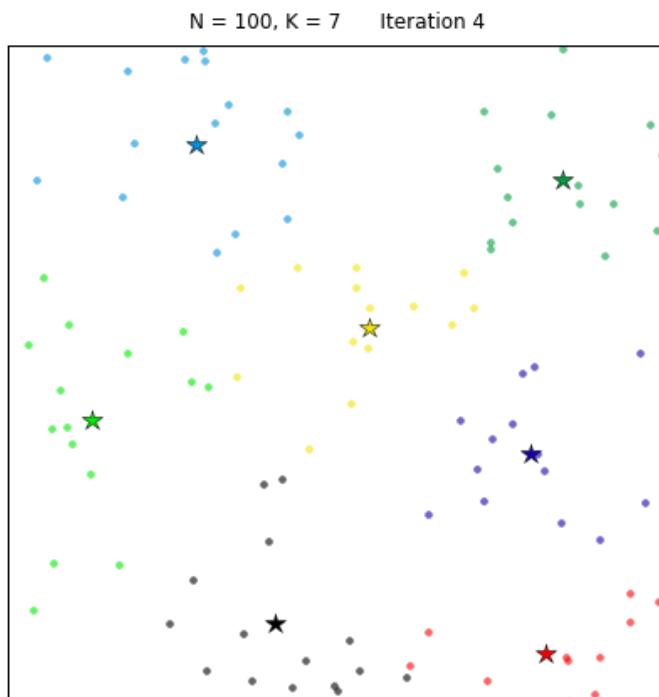
22     return (set([tuple(a) for a in mu]) == set([tuple(a) for a in oldmu]))
23
24 def find_centers(X, K):
25     # Initialize to K random centers
26     oldmu = random.sample(X, K)
27     mu = random.sample(X, K)
28     while not has_converged(mu, oldmu):
29         oldmu = mu
30         # Assign all points in X to clusters
31         clusters = cluster_points(X, mu)
32         # Reevaluate centers
33         mu = reevaluate_centers(oldmu, clusters)
34     return(mu, clusters)

```

Clustering in action

Let's see the algorithm in action! For an ensemble of 100 random points on the plane, we set the k-means function to find 7 clusters. The code converges in 7 iterations after initializing with random centers. In the following plots, dots correspond to the target data points X and stars represent the centroids of the clusters. Each cluster is distinguished by a different color.

(https://datasciencelab.files.wordpress.com/2013/12/p_n100_k7.gif)



The initial configuration of points for the algorithm is created as follows:

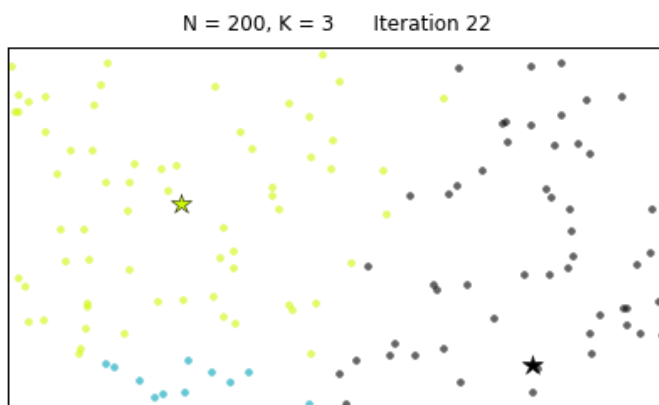
```

1 import random
2
3 def init_board(N):
4     X = np.array([(random.uniform(-1, 1), random.uniform(-1, 1)) for i in range(N)])
5     return X

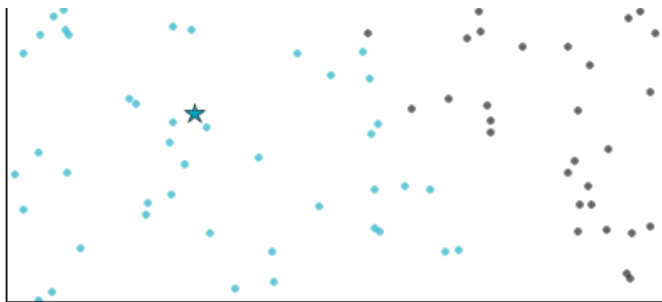
```

For a configuration with twice as many points and a target of 3 clusters, often the algorithm needs more iterations to converge.

(https://datasciencelab.files.wordpress.com/2013/12/p_n200_k3.gif)



Obviously, an ensemble of randomly generated points does not possess a natural cluster-like structure. To make things slightly more tricky, we want to modify the function that generates our initial data points to output a more interesting structure. The following routine constructs a specified number of Gaussian distributed clusters with random variances:

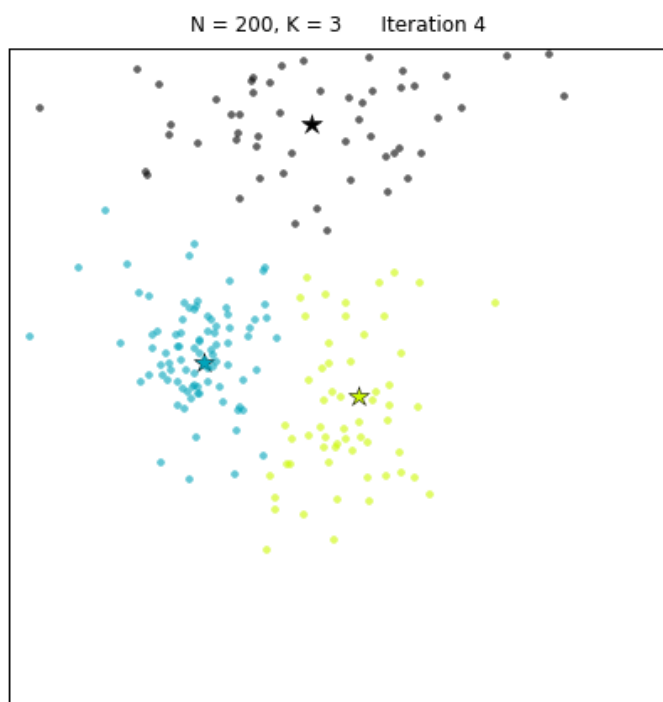


```

1  def init_board_gauss(N, k):
2      n = float(N)/k
3      X = []
4      for i in range(k):
5          c = (random.uniform(-1, 1), random.uniform(-1, 1))
6          s = random.uniform(0.05,0.5)
7          x = []
8          while len(x) < n:
9              a, b = np.array([np.random.normal(c[0], s), np.random.normal(c[1], s)])
10             # Continue drawing points from the distribution in the range [-1,1]
11             if abs(a) < 1 and abs(b) < 1:
12                 x.append([a,b])
13             X.extend(x)
14     X = np.array(X)[:N]
15     return X

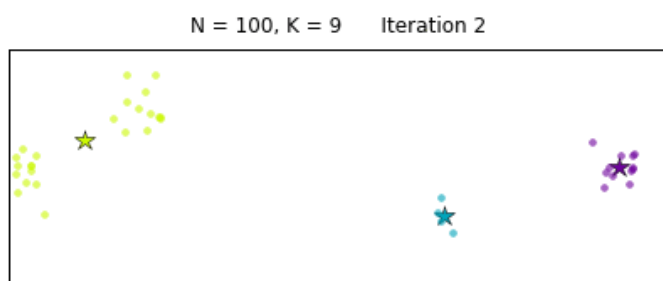
```

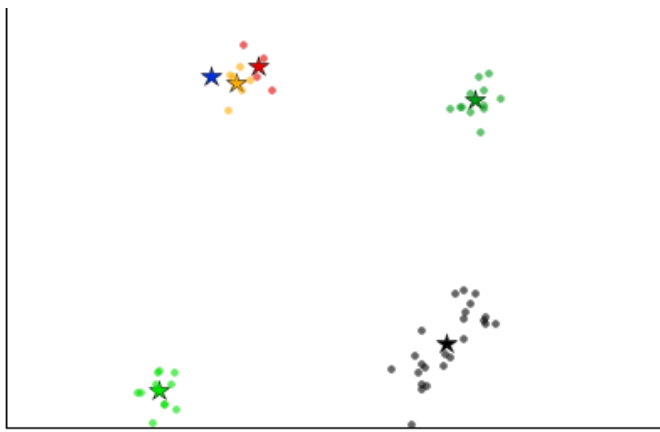
Let us look at a data set constructed as `X = init_board_gauss(200,3)`: 7 iterations are needed to find the 3 centroids.



https://datasciencelab.files.wordpress.com/2013/12/p_n200_k3_g.gif

If the target distribution is disjointedly clustered and only one instantiation of Lloyd's algorithm is used, the danger exists that the local minimum reached is not the optimal solution. This is shown in the example below, where initial data using very peaked Gaussians is constructed:

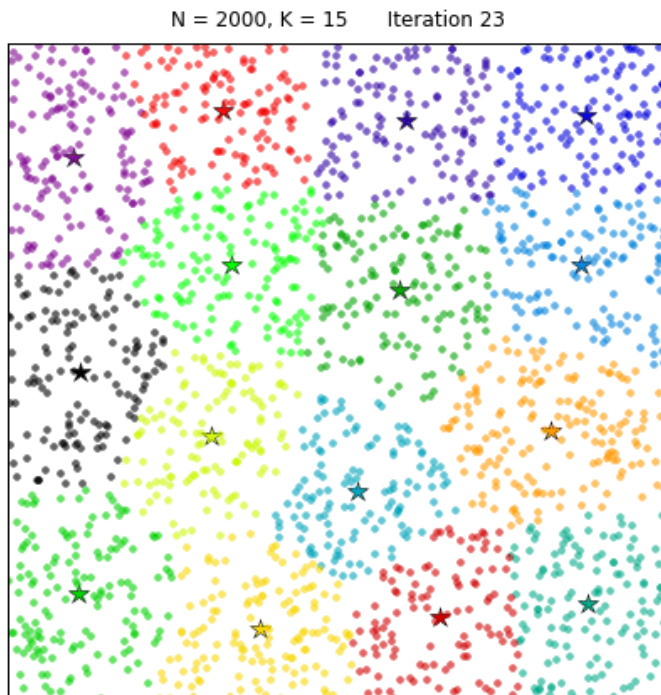




(https://datasciencelab.files.wordpress.com/2013/12/p_n100_k9_g.gif)

The yellow and black stars serve two different clusters each, while the orange, red and blue centroids are cramped within one unique blob due to an unfortunate random initialization. For this type of cases, a cleverer election of initial clusters should help.

To finalize our [table-top experiment](https://datasciencelab.wordpress.com/category/experiments/) (<https://datasciencelab.wordpress.com/category/experiments/>) on k-means clustering, we might want to take a look at what happens when the original data space is densely populated:



(https://datasciencelab.files.wordpress.com/2013/12/p_n2000_k15_g.gif)

The k-means algorithm induces a partition of the observations corresponding to a [Voronoi tessellation](http://en.wikipedia.org/wiki/Voronoi_diagram) (http://en.wikipedia.org/wiki/Voronoi_diagram) generated by the K centroids. And it is certainly very pretty!

Table-top data experiment take-away message

Lloyd's two-step implementation of the k-means algorithm allows to cluster data points into groups represented by a centroid. This technique is employed in many facets of machine learning, from unsupervised learning algorithms to dimensionality reduction problems. The general clustering problem is NP hard, but the iterative procedure converges always, albeit to a local minimum. Proper initialization of the centroids is important. Additionally, this algorithm does not supply information as to which K for the k-means is optimal; that has to be found out by alternative methods.

Update: We explore the gap statistic as a method to determine the optimal K for clustering in this post: [Finding the \$K\$ in K-Means Clustering](https://datasciencelab.wordpress.com/2013/12/27/finding-the-k-in-k-means-clustering/) (<https://datasciencelab.wordpress.com/2013/12/27/finding-the-k-in-k-means-clustering/>) and the method: [Selection of \$K\$ in K-means Clustering, Reloaded](#)

$f(K)$ (<https://datasciencelab.wordpress.com/2014/01/21/selection-of-k-in-k-means-clustering-reloaded/>).

Update: For a proper initialization of the centroids at the start of the k-means algorithm, we implement the [improved k-means++ seeding procedure](https://datasciencelab.wordpress.com/2014/01/15/improved-seeding-for-clustering-with-k-means/) (<https://datasciencelab.wordpress.com/2014/01/15/improved-seeding-for-clustering-with-k-means/>).

About these ads (<https://wordpress.com/about-these-ads/>)

You May Like

?

- 1.



Share!

 (<https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/?share=twitter&nb=1>)

 16 (<https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/?share=facebook&nb=1>)

 (<https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/?share=google-plus-1&nb=1>)

()

Related

Selection of K in K-means Clustering, Reloaded
(<https://datasciencelab.wordpress.com/2014/01/21/selection-of-k-in-k-means-clustering-reloaded/>)
In "Experiments"

Finding the K in K-Means Clustering
(<https://datasciencelab.wordpress.com/2014/01/21/finding-the-k-in-k-means-clustering/>)
In "Experiments"

Improved Seeding For Clustering With K-Means++
(<https://datasciencelab.wordpress.com/2014/01/15/improved-seeding-for-clustering-with-k-means/>)
In "Experiments"

Written by [datasciencelab](https://datasciencelab.wordpress.com/author/datasciencelab/) (<https://datasciencelab.wordpress.com/author/datasciencelab/>) — Posted in [Experiments](https://datasciencelab.wordpress.com/category/experiments/) (<https://datasciencelab.wordpress.com/category/experiments/>) — Tagged with [clustering](https://datasciencelab.wordpress.com/tag/clustering/) (<https://datasciencelab.wordpress.com/tag/clustering/>), [k-means](https://datasciencelab.wordpress.com/tag/k-means/) (<https://datasciencelab.wordpress.com/tag/k-means/>), [Lloyd's algorithm](https://datasciencelab.wordpress.com/tag/lloyds-algorithm/) (<https://datasciencelab.wordpress.com/tag/lloyds-algorithm/>), [python](https://datasciencelab.wordpress.com/tag/python/) (<https://datasciencelab.wordpress.com/tag/python/>), [unsupervised learning](https://datasciencelab.wordpress.com/tag/unsupervised-learning/) (<https://datasciencelab.wordpress.com/tag/unsupervised-learning/>)

35 comments




DECEMBER 12, 2013 - 11:36 PM

[mystatcastle](http://mystatcastle.wordpress.com) (<http://mystatcastle.wordpress.com>)

Reblogged this on [mystatcastle](http://mystatcastle.wordpress.com/2013/12/12/clustering-with-k-means-in-python/) (<http://mystatcastle.wordpress.com/2013/12/12/clustering-with-k-means-in-python/>) and commented:

thanks for sharing, save for future

 (javascript:void(0))

Follow "The Data Science Lab"

Get every new post delivered to your Inbox.

Join 392 other followers

Enter your email address

Sign me up

Build a website with
WordPress.com



Pete Colligan (@linuxster) (<http://twitter.com/linuxster>)

hi. would you offer python notebook version of this?

DECEMBER 27, 2013 - 2:40 PM



datasciencelab (<https://datasciencelab.wordpress.com>)

Sure! I just need to set up a convenient way to share. I guess the notebook viewer <http://nbviewer.ipython.org/> (<http://nbviewer.ipython.org/>) is the easiest? Will look into it.

DECEMBER 27, 2013 - 2:58 PM



Pete Colligan (@linuxster) (<http://twitter.com/linuxster>)

Hello,

You can export your ipython notebook markup and upload it to github gist. Then reference the gist URL from the <http://nbviewer.ipython.org/> (<http://nbviewer.ipython.org/>) you mentioned above.

you can also just email me (linuxster at gmail dot com) source tarball or post your source on github and I can clone.

**I have put an example of ILP tutorial notebooks here: <http://nbviewer.ipython.org/gist/linuxster/7295256> (<http://nbviewer.ipython.org/gist/linuxster/7295256>)

currently I am focusing on the data characteristics that feed the algorithm, especially in the case where I might automate as much as possible and therefore must take into account pre-processing (missing data, outliers, encoding, feature selection, etc)

maybe you have experience here already?

1. I am interested in exploring the effects of L1 and L2 norm on the cluster output. in your example you use the L2 norm but since Kmeans is sensitive to outliers maybe L1 norm is better?
2. also I want to explore whether I can use the mode instead of mean for the centroid recalculation in step two of LLoyds algorithm. again, something to minimize impact of outliers.
3. also if I have mixed data (categorical and numerical) that I would cluster together what is the best approach for pre-processing the data? I have heard of some folks encoding the categorical data into a binary vector format to support the distance calculations but would like to get other opinions here as well.

just saw there is a follow up post to this one on k selection. will read.

DECEMBER 27, 2013 - 3:31 PM



datasciencelab (<https://datasciencelab.wordpress.com>)

Thanks! I will set up my github/gist soon and will keep you posted.

Your discussion on L1/L2 norm is very interesting. What I've read so far always uses L2 to measure the intracluster variance, but exploring L1 is surely worthwhile. Re: mode versus mean, I guess it depends on the distribution of your data. My play data just follows normal distribution or random, thus I don't think that the mode brings much more than the mean. It'd be interesting to see what happens when data follows other shapes though. Regarding mixed data, I can't offer much insight there. What kind of categorical data do you have in mind?

DECEMBER 27, 2013 - 3:00 PM



Pete Colligan (@linuxster) (<http://twitter.com/linuxster>)

sorry. forgot to reference the notebook I refer to I cloned from nipun batra.

DECEMBER 28, 2013 - 12:22 PM

 [Pete Colligan \(@linuxster\)](#) (<http://twitter.com/linuxster>)

I have data from a customer which includes retail store characteristics. They would like to use many store attributes to help them find new store "groupings" that they can use for business planning activities. There are about 50 attributes per store and data is numerical, ordinal, and categorical in nature. (size, nearest competitor, demographic info, sales, market segment, etc.). I would like to first:

1. get an idea of which attributes account for the most variance in the data. PCA is ok but I get a bit lost mapping from one basis to the other. (ex. which PCA component contains which features from my dataset). Goal here is some dimensionality reduction since I know some of my features are collinear.
2. after certain "redundant" information is removed, I think I should look at the data distribution of each feature to get an idea of the "nature" of the data. not quite sure here any hard and fast rules. Ex. Normalization, Regularization?
3. then I would go for clustering algorithm. At the moment I like Kmeans and agglomerative hierarchical.

what do you think about that thought process?


note: Kmeans will force every point to a cluster. Outliers are not ALWAYS bad in this use case (some stores like "super stores" are important to keep and might be an own cluster) so I don't want to always just remove them prior to running the algorithm.

DECEMBER 28, 2013 - 5:27 PM

 [datasciencelab](https://datasciencelab.wordpress.com) (<https://datasciencelab.wordpress.com>)

I see that your problem is another kind of beast, as you probably need to start with a multivariate analysis to determine the relative importance of each attribute (your step 1, for which you could do some ANOVA) before you can proceed to clustering on them. Your strategy seems correct, and due to the complexity of the problem, I'm sure you'll find many interesting steps to complete along the way. Good luck!

FEBRUARY 25, 2014 - 7:44 PM

 Mike Files

Reblogged this on [Big Data . . .](http://mikefiles.wordpress.com/2014/02/25/clustering-with-k-means-in-python/) (<http://mikefiles.wordpress.com/2014/02/25/clustering-with-k-means-in-python/>) and commented:

Module 4: Advanced Analytics: K-Means Clustering

APRIL 11, 2014 - 7:36 AM


Pingback: [Grouping countries according to flag similarity - Virostatiq](http://virostatiq.com/grouping-countries-according-flag-similarity/) (<http://virostatiq.com/grouping-countries-according-flag-similarity/>)

JULY 1, 2014 - 4:22 PM

 [guido](http://dropbox.com) (<http://dropbox.com>)

do not use `i[0]` or `mu[]`
instead `i,m enumerate(mu)` directly

JULY 26, 2014 - 9:19 AM

 [datasciencelab](https://datasciencelab.wordpress.com) (<https://datasciencelab.wordpress.com>)

Thanks for pointing out the superior solution with `enumerate`, Guido!

JULY 18, 2014 - 3:36 AM

Pingback: [@TKMafioso "Family Struggles" Official Video #NEWMUSIC #IEMUSICGANG | #iRapindIE](http://www.irapindie.com/newmusic/tkmafioso-family-) (<http://www.irapindie.com/newmusic/tkmafioso-family->



JULY 24, 2014 - 8:43 AM
nico

Hi there,
big thanks for all of that, seems to be exactly what I was looking for. I am not that new to python, but new to clustering but i guess I will find my way to use the implementations on my data.
One thing I was wondering is what would be the best way to keep track of the sample id's, which is of importance in my research? Can someone point me in a direction?
Thanks



AUGUST 22, 2014 - 4:25 AM
Lars

Thank you for this excellent set of examples. K-Means and Lloyd's Algorithm are probably what I'm looking for. I'm trying to investigate a cluster problem that has a twist. Specifically, I would like to find good locations for public schools, based on the geographical distribution of students. I can estimate the X-Y coordinates of the students on a city map, and could do a simple cluster analysis. However the problem is complicated by the fact that the city has many internal barriers (waterways, railway tracks, fenced expressways, etc) which divide the city into many smaller areas, and which can only be crossed at a limited number of points. This means that two points whose x-y coordinates might be close, could in fact be separated by a barrier that forces a much greater travel distance.
The distance between a point in area A, and another point in Area B, is in fact the sum of their distances to the bridge point across the barrier. (rather than simply the distance calculated from their x-y coordinates.) That is fairly straight forward. My thinking is that I should try to intercept the cluster processing at some midpoint and substitute the practical distance for the distance calculated from the x-y coordinates, and then proceed with the analysis.
I have some experience with Python, but I have no formal training in Cluster Analysis.
My questions are:
1. Is it possible to intercept the processing and substitute different distance values.
2. Does this type of problem have a standard solution, and I simply haven't found it.
3. Do you recommend some other approach.
Thanks,



AUGUST 22, 2014 - 4:11 PM
vineeth

Thanks for the example. How can i implement k-means algorithm for finding severe and normal facial palsy using python.
Thanks in advance.

SEPTEMBER 30, 2014 - 4:52 PM

Pingback: [Machine Learning and K-Means Clustering on Zillabyte - Zillablog \(http://blog.zillabyte.com/2014/09/30/machine-learning-k-means-clustering-zillabyte/\)](http://blog.zillabyte.com/2014/09/30/machine-learning-k-means-clustering-zillabyte/)



OCTOBER 2, 2014 - 9:09 AM
[Ashish Dutt \(http://www.edumine.wordpress.com\)](http://www.edumine.wordpress.com)

Reblogged this on [Educational Data Mining \(EDM\) and You! \(http://edumine.wordpress.com/2014/10/02/clustering-with-k-means-in-python/\)](http://edumine.wordpress.com/2014/10/02/clustering-with-k-means-in-python/) and commented:
Very interesting read.



NOVEMBER 14, 2014 - 3:58 AM
joon

Thanks a lot for the post. It seems closing `)` is missing from the line: `return (set([tuple(a) for a in mu]) == set([tuple(a) for a in oldmu]) in def has_converged(mu, oldmu).`



MARCH 3, 2015 - 6:08 PM
[muni \(http://www.50states.com/abbreviations.htm#U3YC79KSwrg\)](http://www.50states.com/abbreviations.htm#U3YC79KSwrg)

Thanks for sharing. Very informative. but, could you please update me with any sample data

MARCH 14, 2015 - 10:06 AM
Pingback: [Distilled News | Data Analytics & R \(http://advanceddataanalytics.net/2015/03/14/distilled-news-36/\)](http://advanceddataanalytics.net/2015/03/14/distilled-news-36/)



MARCH 15, 2015 - 1:26 AM
[Anh Le \(http://bigsonata.com\)](http://bigsonata.com)

Reblogged this on [Big Sonata \(http://bigsonata.com/2015/03/15/clustering-with-k-means-in-python/\)](http://bigsonata.com/2015/03/15/clustering-with-k-means-in-python/).

MARCH 15, 2015 - 2:26 PM
Pingback: [Clustering With K-Means in Python | d@n3n | Sc... \(http://www.scoop.int/d-n3n/p/4039186582/2015/03/15/clustering-with-k-means-in-python/\)](http://www.scoop.int/d-n3n/p/4039186582/2015/03/15/clustering-with-k-means-in-python/)



MARCH 23, 2015 - 3:43 PM
[fnokeke \(http://onetorch.wordpress.com\)](http://onetorch.wordpress.com)

Reblogged this on [onetorch \(https://onetorch.wordpress.com/2015/03/23/clustering-with-k-means-in-python/\)](https://onetorch.wordpress.com/2015/03/23/clustering-with-k-means-in-python/) and commented:
Straightforward tutorial on K-Means Clustering.



MARCH 24, 2015 - 12:17 AM
[Giri Kuncoro \(http://pentrioloquist.wordpress.com\)](http://pentrioloquist.wordpress.com)

Reblogged this on [Giri Kuncoro \(https://pentrioloquist.wordpress.com/2015/03/24/clustering-with-k-means-in-python/\)](https://pentrioloquist.wordpress.com/2015/03/24/clustering-with-k-means-in-python/).



MARCH 30, 2015 - 10:01 PM
Anonymous

Hello, how to visualize the iterative clustering steps with python? Thank you.

APRIL 29, 2015 - 1:39 PM
Pingback: [Кластеризация с помощью метода k-средних на Python | \(http://datareview.info/article/klasterizatsiya-s-pomoshhyu-metoda-k-srednih-na-python/\)](http://datareview.info/article/klasterizatsiya-s-pomoshhyu-metoda-k-srednih-na-python/)



MAY 4, 2015 - 4:30 AM
kevin

Do you know why I am getting the error “global name ‘zeros’ is not defined”?

Thanks

Kevin



MAY 22, 2015 - 4:54 PM

Jiajun

Hi, Thanks for this amazing example.

But I really struggle to understand:

```
bestmukey = min([(i[0], np.linalg.norm(x - mu[i[0]])) \
for i in enumerate(mu)], key = lambda t: t[1])[0]
```

I know what it is trying to achieve: outputting the key (index) of mu, which has the shortest distance to x. I think my struggle is to understand the use of min() and enumerate(). Is it possible you can rewrite it in a normal loop?

Appreciate it!!

J



JUNE 20, 2015 - 6:21 PM

[ly08096 \(http://ly08096.wordpress.com\)](http://ly08096.wordpress.com)

Reblogged this on [Yi's Personal Notes \(https://ly08096.wordpress.com/2015/06/20/clustering-with-k-means-in-python/\)](https://ly08096.wordpress.com/2015/06/20/clustering-with-k-means-in-python/).



AUGUST 13, 2015 - 12:57 PM

[richarddunks \(http://blog.datapolitan.com\)](http://blog.datapolitan.com)

Reblogged this on [Datapolitan \(http://blog.datapolitan.com/2015/08/13/clustering-with-k-means-in-python/\)](http://blog.datapolitan.com/2015/08/13/clustering-with-k-means-in-python/) and commented:

A great overview of k-Means clustering in Python with good visual explanations.



OCTOBER 27, 2015 - 8:54 AM

phil

yes, can you explain that bestmukey line for non python users

the goal is to explain KMeans, so why do you use such a non standard syntax, come on



NOVEMBER 29, 2015 - 3:51 PM

Master

Does anyone know how to label the dots in the cluster ? I want to label some dots in the cluster, but I don't know how.



NOVEMBER 30, 2015 - 9:11 AM

nico

@Master, I did asked sort of that question a while ago (July 24, 2014), without an answer.

After the clustering is done, my workaround consist of comparing each element in the 'cluster' dictionary to the input list where the data-to-id connection is known. The id gets transferred to a similar structured dictionary like 'clusters', which in the end is holding all the id's. Might be far from elegant, but makes annotation via matplotlib easy...



DECEMBER 4, 2015 - 8:14 AM

[mannyglover \(http://gravatar.com/mannyglover\)](http://gravatar.com/mannyglover)

I really appreciate this post, but can you post code to show how to visualize? I know it can't be uber-complicated, but it would be nice to see the code that you used to visualize the data. Thanks.

[A Peek at the ORCID Registry Public Data File \(https://datasciencelab.wordpress.com/2013/12/05/a-peek-at-the-orcid-registry-public-data-file/\)](https://datasciencelab.wordpress.com/2013/12/05/a-peek-at-the-orcid-registry-public-data-file/)

[On MOOCs and The Audacious \(https://datasciencelab.wordpress.com/2013/12/17/on-moocs-and-the-audacious/\)](https://datasciencelab.wordpress.com/2013/12/17/on-moocs-and-the-audacious/)

