Documentation

Here are the functions in the code and what they do

distance(m, p):

distance is a function that return the euclidean distance between two points.

```
In [5]: distance(cardata[1], cardata[65])
Out[5]: 2.449489742783178
```

avg(L):

avg is a function that takes as an input a cluster of points and returns the average of these points (the new centroid)

```
In [7]: avg(cardata[1:100])
Out[7]: [1.0, 1.0, 2.393939393939394, 1.9292929292929293, 2.0, 2.0]
```

random_init(data, K):

random_init is a function that takes data and number of clusters (K) and return K random points of the data as the centroids of the clusters.

```
In [8]: random_init(cardata, 4)
Out[8]:
[[4, 3, 2, 3, 2, 1],
[2, 4, 2, 3, 3, 2],
[1, 4, 2, 1, 3, 1],
[4, 3, 3, 1, 1, 2]]
```

kmeans(data, K):

The main function that implements the k means algorithm. it takes as an input the data and number of clusters K and returns the centroids and the clusters assignment list for the points.

```
In [12]: centroids, clusters = kmeans(cardata, 4)
In [13]: centroids
Out[13]:
[[1.411764705882353, 2.5882352941176472, 1.6470588235294117, 2.0,
2.0, 2.0],
[3.5625, 2.375, 1.5625, 2.0, 2.0, 2.0],
[2.625, 3.5625, 3.4375, 2.0, 2.0, 2.0],
```

```
[2.4666666666666668, 1.3999999999999999, 3.4666666666666668, 2.0,
2.0, 2.0]]
In [14]: len(clusters)
Out[14]: 1728
In [15]: clusters[0:10] # the cluster assignment of the first 10
points.
Out[15]: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

error(data, clusters, K, classified_data, classes):

error is a function that takes as an input the numerical data, the clusters assignment list, the number of clusters K, the raw categorical data and the list of classes and return the count of errors in classification, the clusters and the ratio of error.

```
In [17]: e, real_clustes, ratio = error(cardata, clusters, 4,
formatted_data, classes)
In [18]: e
Out[18]: 518
In [19]: ratio
Out[19]: 29.976851851851855
```

At the end of the code there are some commented lines of code that you can uncomment before running the code and changing the variables 'lower_limit' and 'upper_limit' to build a range of cluster numbers starting from lower limit to upper limit. The code will cluster the data based on all these clusters, calculate the error for each case and then plot the number of clusters on x axis against the error on the y axis. You need to have 'matplotlib' library for this plotting to happen. (there is an example in the discussion).

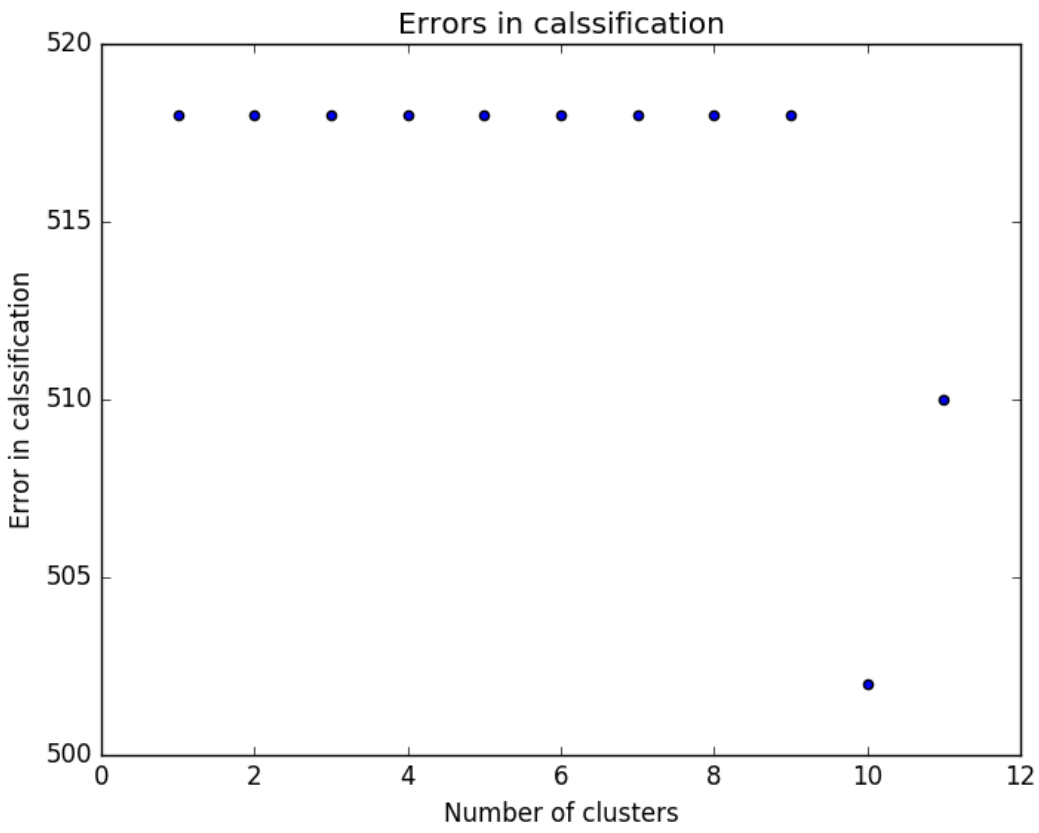
Running the script

- **Requirements**
  - Python 2.7
  - numpy
  - matplotlib (only for the last optional commented part)
- Navigate to the directory that contains the script 'kmeans.py' through the command line.
- Type **python kmeans.py**

```
amr@amr-Laptop:~/machine_learning/kmeans$ python kmeans.py
Enter the input data File Name: car.data
Enter the number of clusters you want: 4
With clustering the data into 4 clusters, we got 518 errors with
classification accuracy of 70.02 %
```

You will be asked to enter the data file name and to specify the number of cluster K you want. K has to be an integer. The file name can be just the name of the file if it is in the same directory as the script (preferable way) or the absolute path.

Discussion



With choosing 4 as number of clusters K, we get error of 518 with accuracy of classification of about 70% almost always despite the random initialization and that is explainable. The data is hugely skewed as we have 1728 instances 1210 of them are classified as ‘unacc’, 384 as ‘acc’, 69 as ‘good’ and 65 as ‘vgood’. From this data and also putting into account that we converted categorical data into numerical data which doesn’t make sense in terms of the euclidean distance between categorical points, we realize that when we choose small number of clusters such as 4 we will always end up with a majority vote inside each cluster in favor of ‘unacc’ class. In other words, all the points are classified ‘unacc’ and so that the error is always the count of the other classes combined 384 + 69 + 65 = 518 because the y are wrongly classified. When we increase the number of clusters starting from 10 clusters for example, we observe a decrease in error and that is because at least one of the clusters gets classified with different class most probably the next prevalent one ‘acc’ so we get a different error. We can see that from the plot shown above.

To sum up, k means is not an ideal algorithm to classify categorical data especially if the data has a huge dominance of a certain class.