k_means

September 16, 2018

1 Importing packaes

```
In [1]: import numpy as np
        from sklearn.datasets import load_iris
        import matplotlib.pyplot as plt

        import copy
        iris = load_iris()
        print(iris.DESCR)
```

```
Iris Plants Database
====================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)
    ============== ==== ==== ======= ===== ====================
```

```
    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML iris datasets.
http://archive.ics.uci.edu/ml/datasets/Iris

The famous Iris database, first used by Sir R.A Fisher

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

References
----------
   - Fisher,R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

2 Dataset

```
In [2]: x = np.transpose(iris.data)
        y = iris.target

        print(x.shape)
        print(y.shape)
```

```
(4, 150)
```

```
(150,)
```

3 Parameters

```
In [3]: samples = x.shape[1]
        dim_samples = x.shape[0]
        k = 3
        epselon = 0.0001
```

4 Initializing the centroids

```
In [4]: new_centroids = (np.repeat(np.max(x, axis = 1) - np.min(x, axis = 1),k).reshape(dim_samp
        centroids = np.zeros(new_centroids.shape)
```

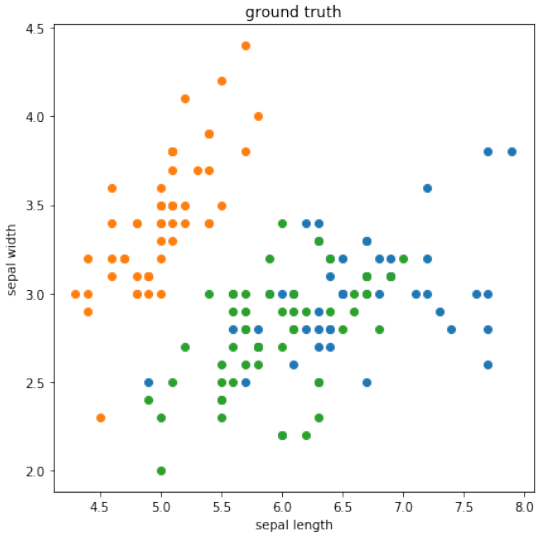
5 Clustering and finding the centroids

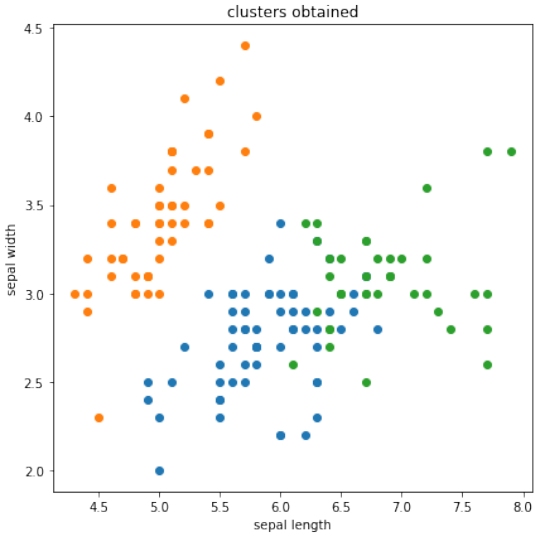
```
In [5]: while(np.sum(np.square(centroids - new_centroids)) > epselon):
            dist = []
            centroids = copy.deepcopy(new_centroids)
        #     finding distance for each sample with each centroid
            for i in range(0,k):
                temp = np.transpose(np.tile(centroids[:,i],samples).reshape(samples , dim_sample
                dist.append(np.sum(np.square(x - temp),axis = 0))
            dist = np.array(dist)
        #     finding the nearest centroid
            dist_min_index = np.argmin(dist, axis = 0)

        #     finding the new centroids
            for i in range(0,k):
                x_temp = []
                x_temp.append(x[:,dist_min_index == i])
                x_temp = np.squeeze(np.array(x_temp),axis = 0)
                if x_temp.size:
                    centroid = np.mean(x_temp, axis = 1)

                    new_centroids[:,i] = centroid
            print("diff: " , np.sum(np.square(centroids - new_centroids)))
```

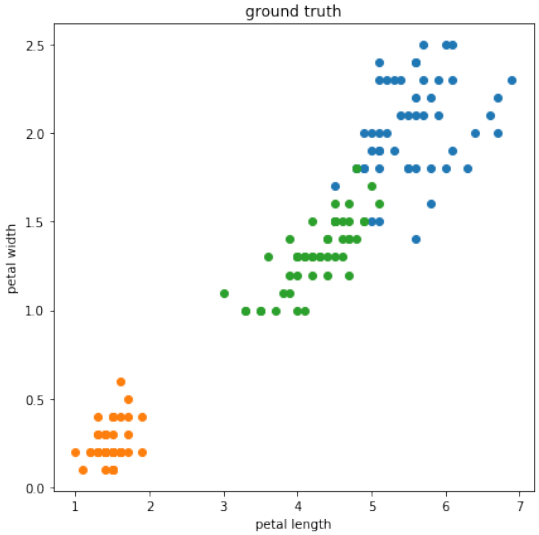
```
diff:  10.026679763960859
diff:  0.8861643006336736
diff:  0.08608828836451975
diff:  0.011589785911915585
diff:  0.003134836274977712
diff:  0.0014561250815201643
diff:  0.0
```

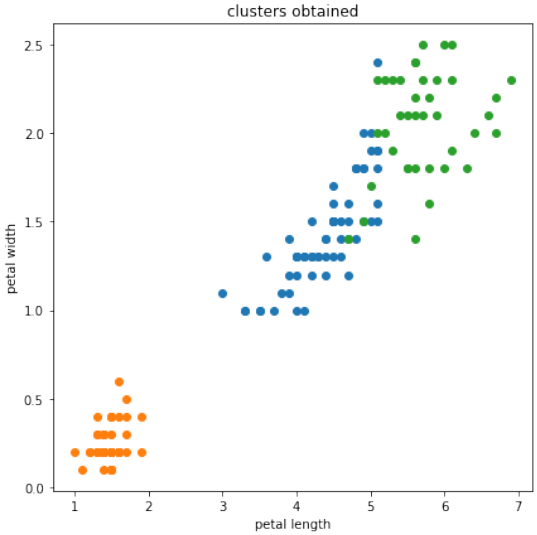
6 Ploting the results for visualisation

In [6]:
```
plt.figure(figsize=(15,15))
plt.subplot(2,2,1)
plt.plot(x[0,y==2],x[1,y==2],'o',x[0,y==0],x[1,y==0],'o', x[0,y==1],x[1,y==1],'o')
plt.title("ground truth")
plt.xlabel("sepal length")
plt.ylabel("sepal width")


plt.subplot(2,2,2)
plt.plot(x[0,dist_min_index==2],x[1,dist_min_index==2],'o',x[0,dist_min_index==1],x[1,di
plt.title("clusters obtained")
plt.xlabel("sepal length")
plt.ylabel("sepal width")

plt.subplot(2,2,3)
plt.plot(x[2,y==2],x[3,y==2],'o',x[2,y==0],x[3,y==0],'o', x[2,y==1],x[3,y==1],'o')
plt.title("ground truth")
plt.xlabel("petal length")
plt.ylabel("petal width")


plt.subplot(2,2,4)
plt.plot(x[2,dist_min_index==2],x[3,dist_min_index==2],'o',x[2,dist_min_index==1],x[3,di
plt.title("clusters obtained")
plt.xlabel("petal length")
plt.ylabel("petal width")


plt.show()

# as these are the clusters , the colors of the clusters in the graphs may not be in ord
```


ground truth
sepal width
sepal length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


clusters obtained
sepal width
sepal length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


ground truth
petal width
petal length
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7


clusters obtained
petal width
petal length
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7