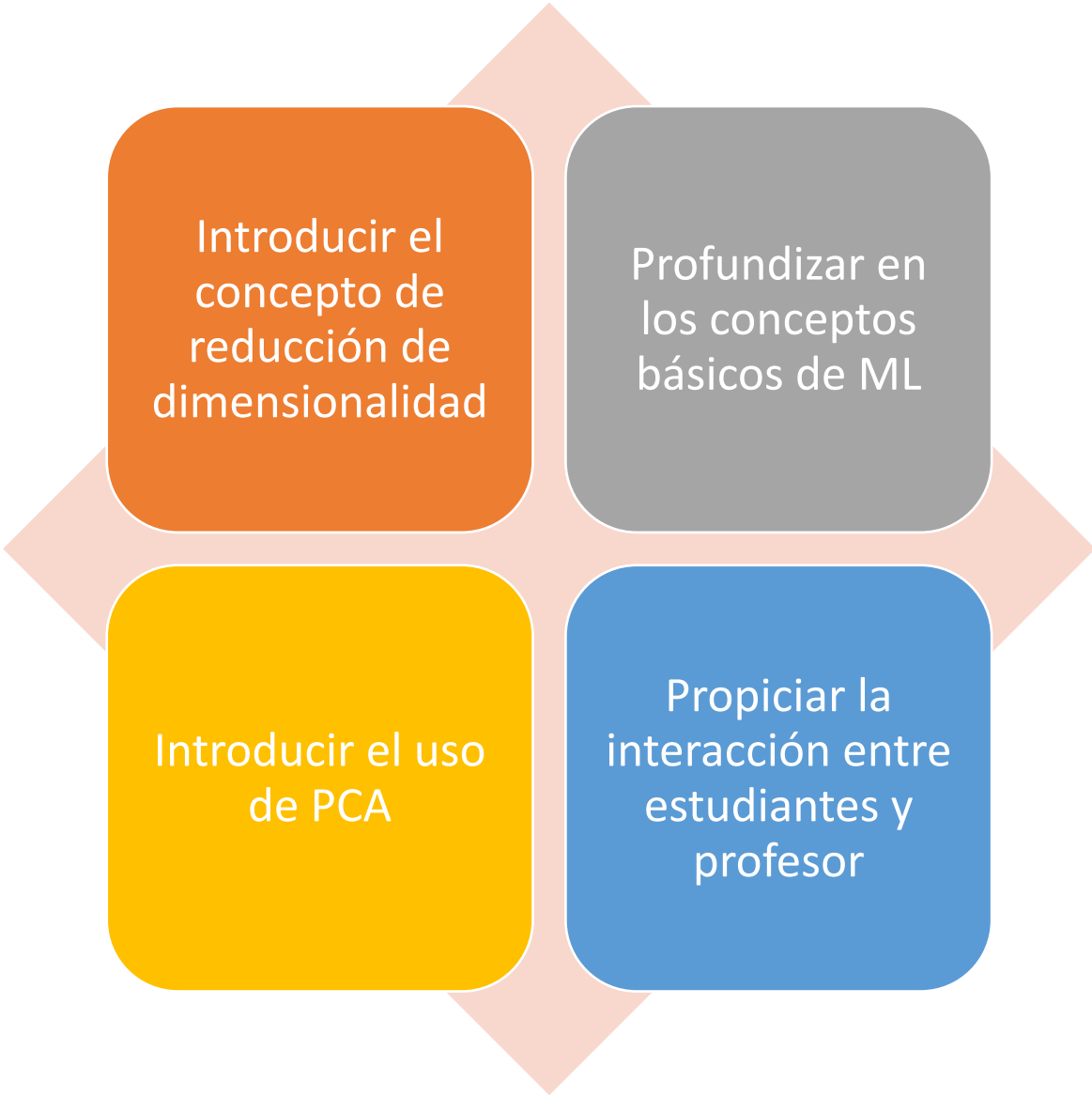
Inteligencia Artificial LFIS419

Clase 7: k-nearest neighbors y kmeans

Profesor: Jorge Arevalo (jorge.arevalo@uv.cl jab@meteo.uv.cl)

Martes 9 de mayo de 2023

Objetivos de la sesión



kNN

- Típicamente para clasificación, pero se puede usar para regresión
- 2 hiperparámetros k y distancia (métrica)
- Regularizar datos
- Requiere mucha memoria

$$\text{Minkowski Distance} = \left(\sum_{i=1}^{n} |x_i - y_i|\right)^{1/p}$$

p=2 -> euclidiana
P=1 -> Manhattan

https://www.ibm.com/topics/knn

kNN

```
from sklearn.neighbors import KNeighborsClassifier
model_name = ‘K-Nearest Neighbor Classifier’
knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric = ‘minkowski’, p=2)
knn_model = Pipeline(steps=[(‘preprocessor’, preprocessorForFeatures), (‘classifier’ , knnClassifier)])
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
```

kMeans

El algoritmo consta de tres pasos:
1. Inicialización: una vez escogido el número de grupos, *k*, se establecen *k* centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.

2. Asignación objetos a los centroides: cada objeto de los datos es asignado a su centroide más cercano.

3. Actualización centroides: se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso.

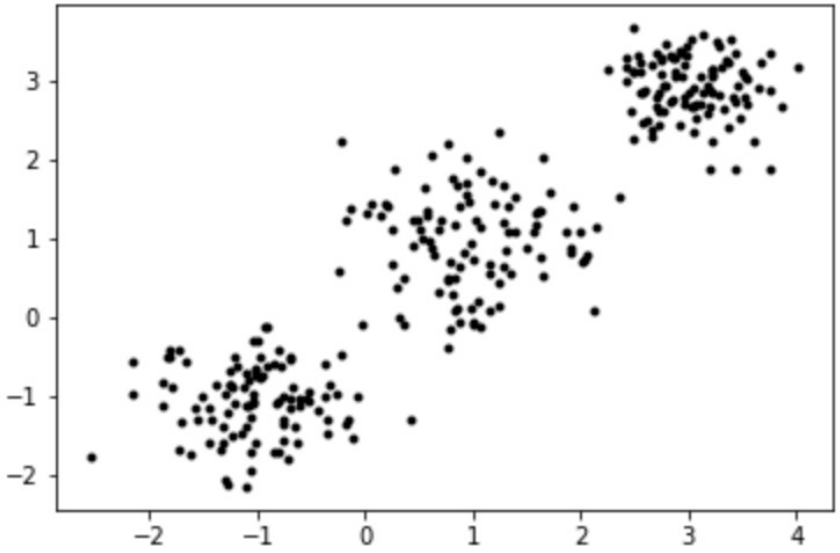
In [1]:
```
%matplotlib inline
```

Importamos las librerías y funciones necesarias

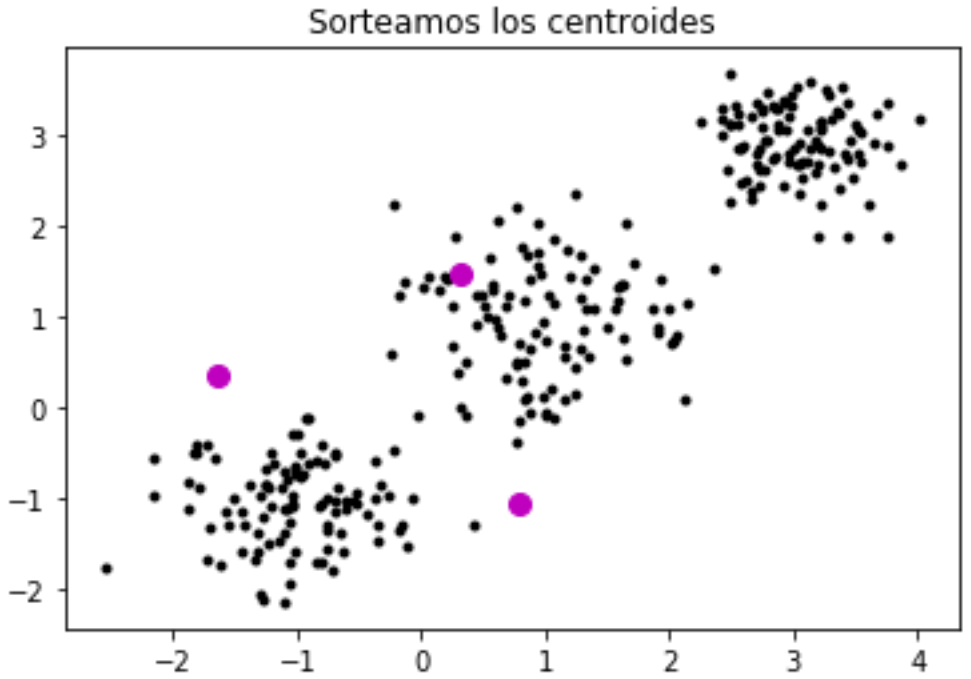
In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

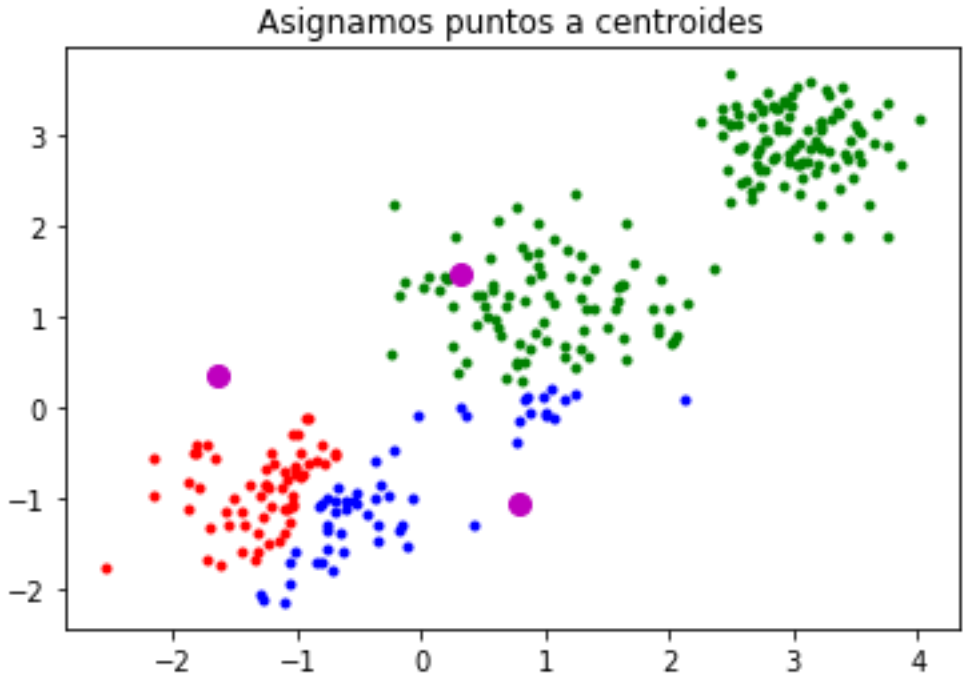
Generamos datos aleatorios 2D para tres clusters y los representamos

In [3]:
```
np.random.seed(7)

x1 = np.random.standard_normal((100,2))*0.6+np.ones((100,2))
x2 = np.random.standard_normal((100,2))*0.5-np.ones((100,2))
x3 = np.random.standard_normal((100,2))*0.4-2*np.ones((100,2))+5
X = np.concatenate((x1,x2,x3),axis=0)

plt.plot(X[:,0],X[:,1],'k.')
plt.show()
```

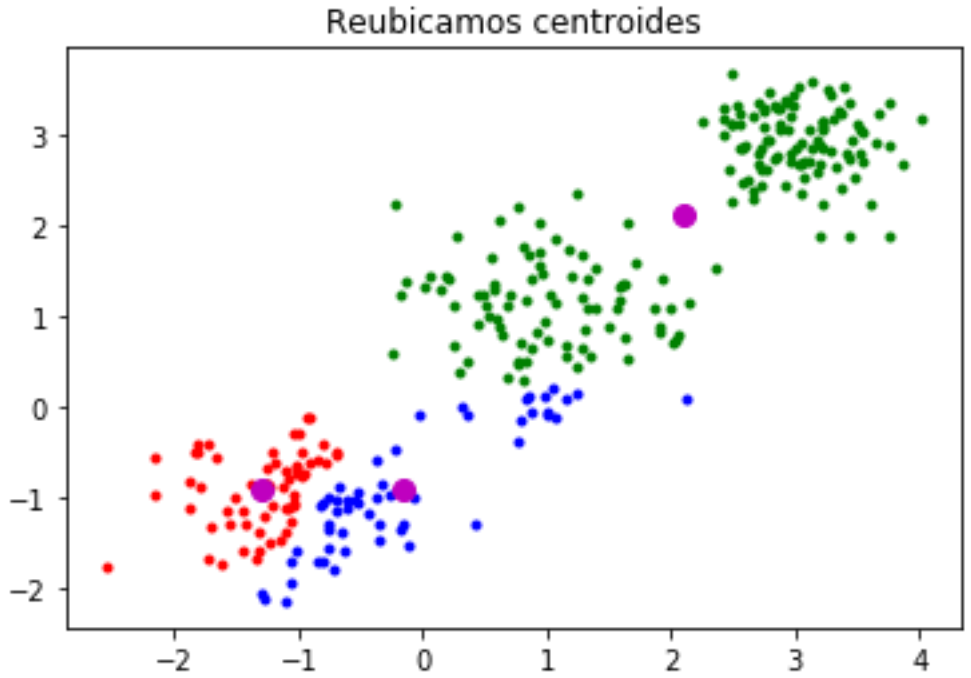


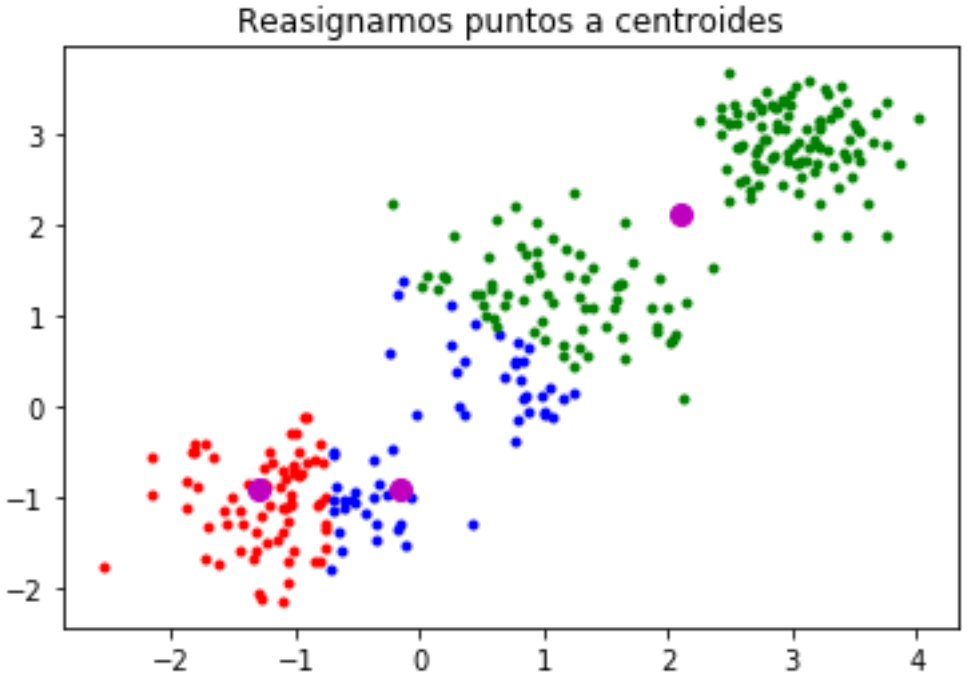
Inicio



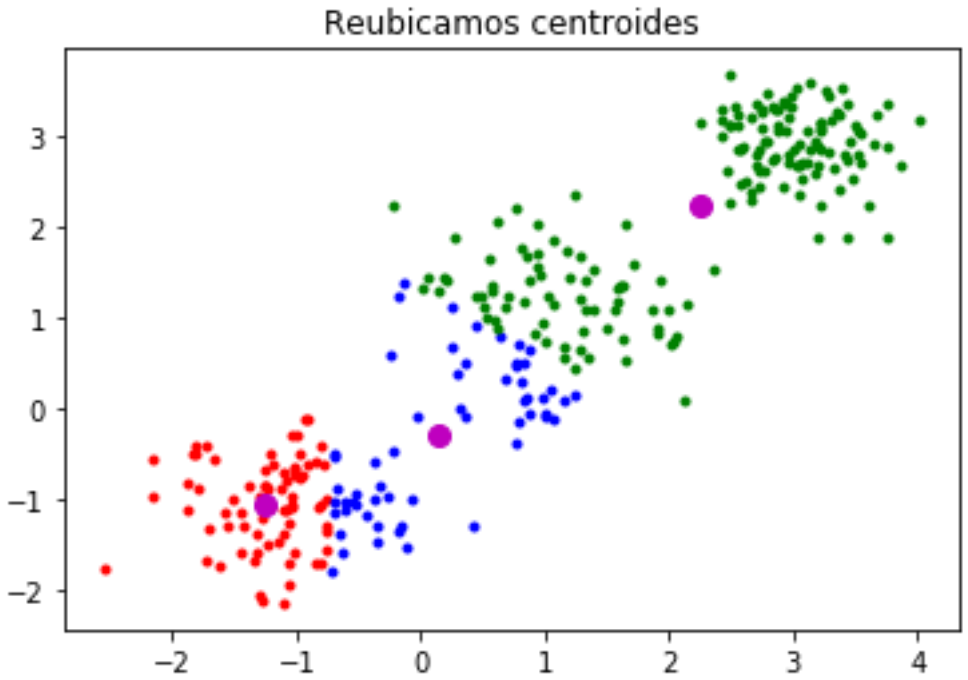

Asignamos puntos a centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

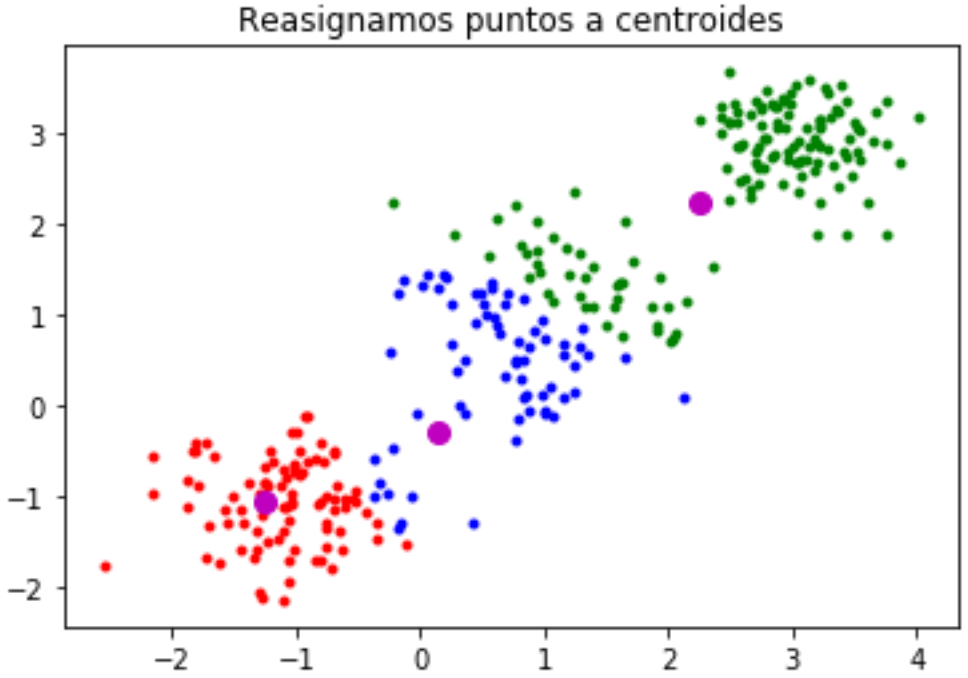
Iteración 1


Reubicamos centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

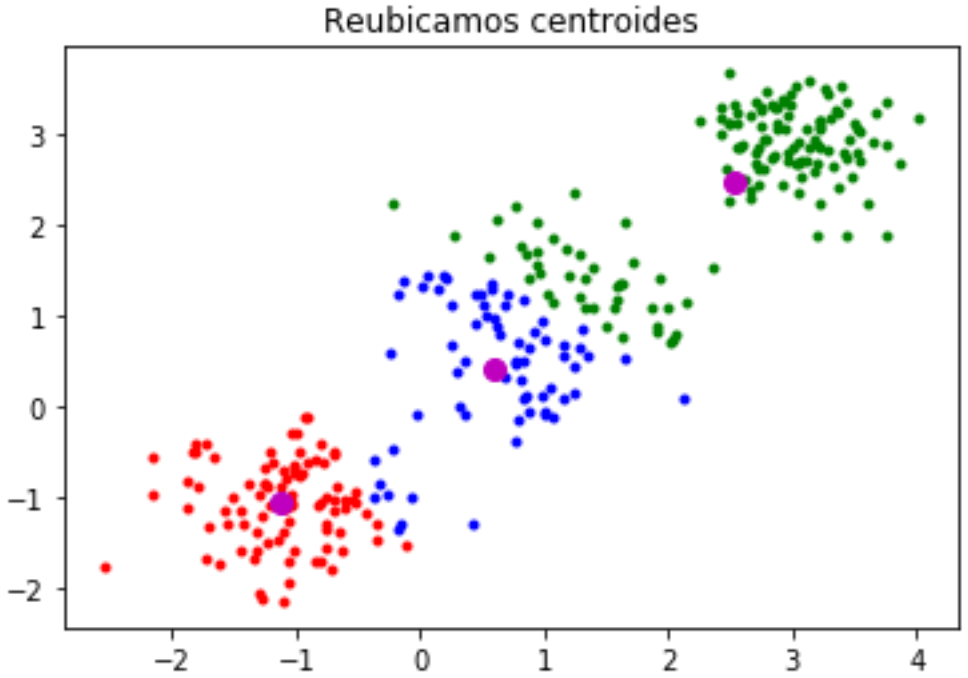

Reasignamos puntos a centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

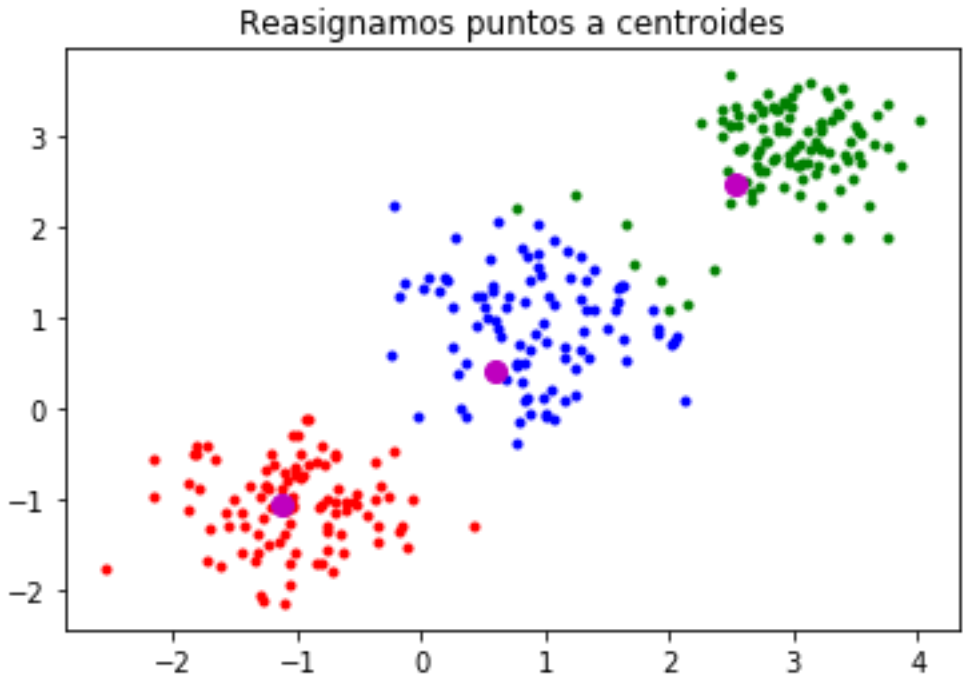
Iteración 2


Reubicamos centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

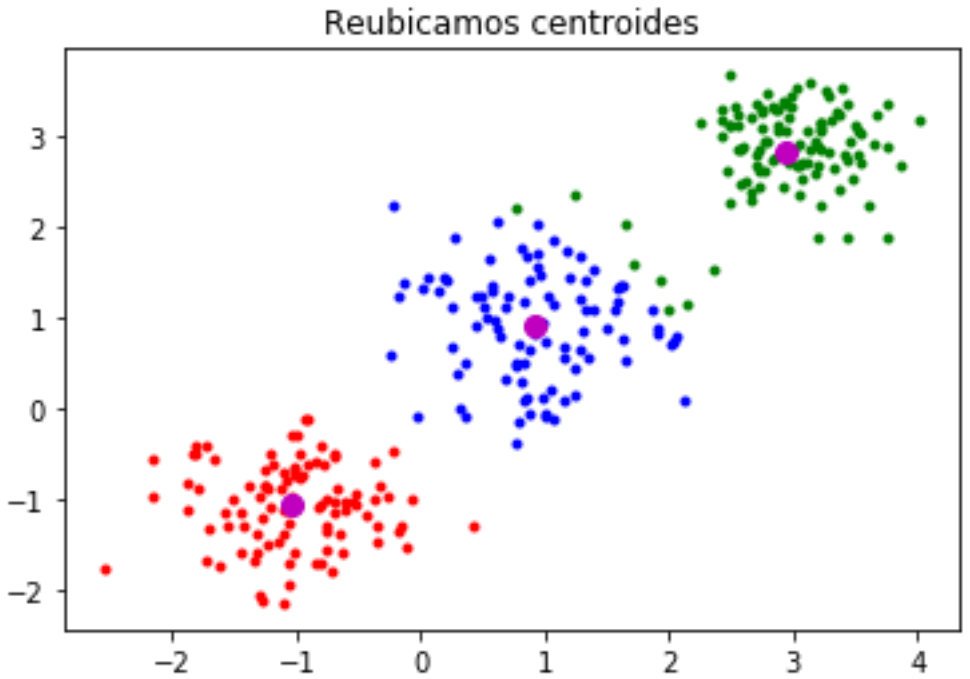

Reasignamos puntos a centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

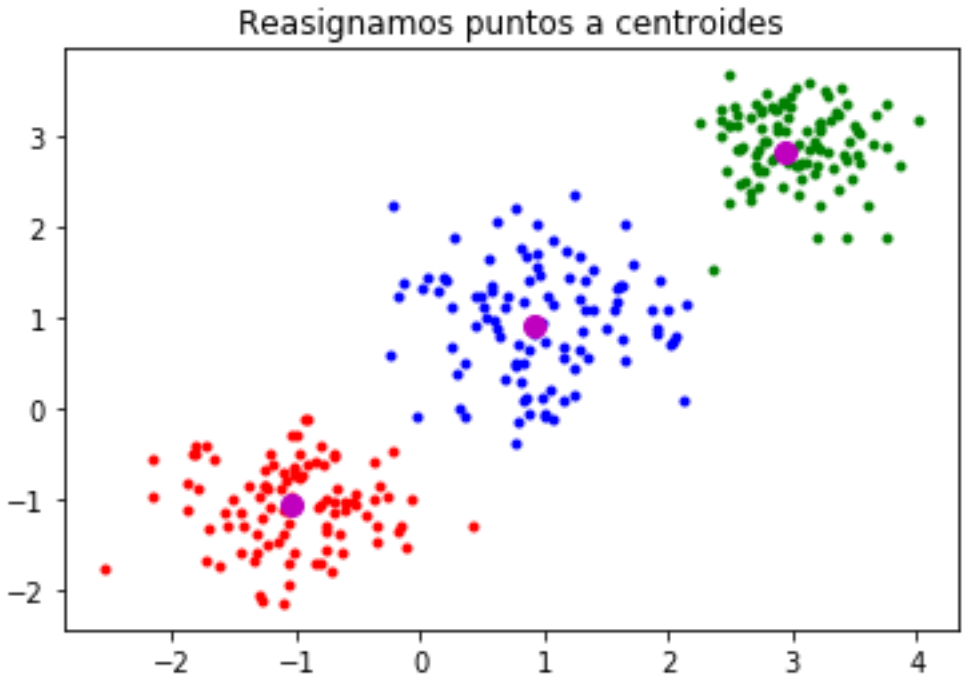
Iteración 3


Reubicamos centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

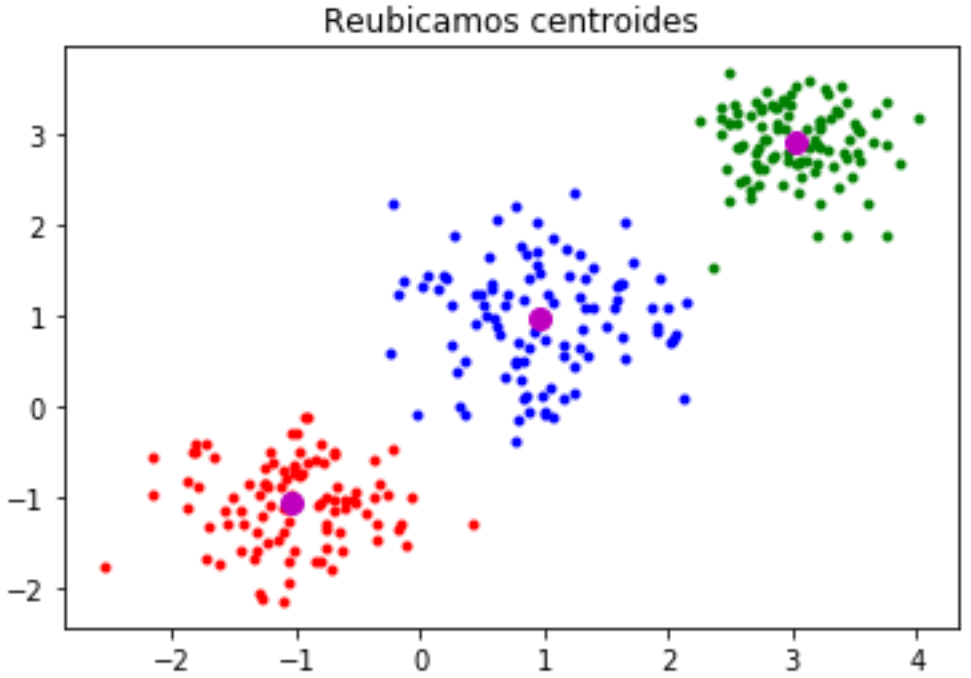

Reasignamos puntos a centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

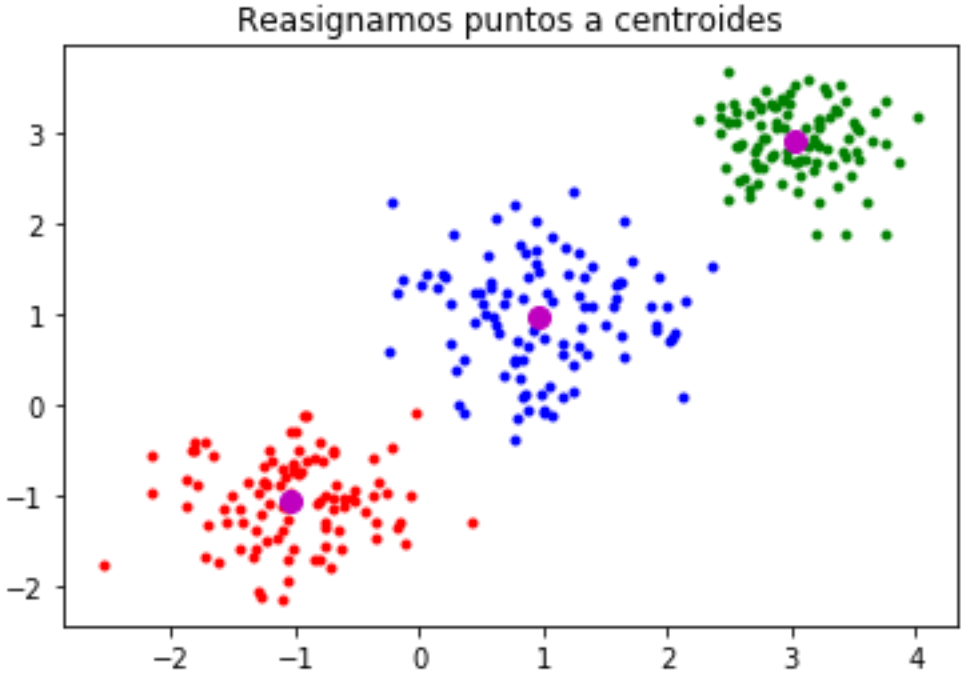
Iteración 4


Reubicamos centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

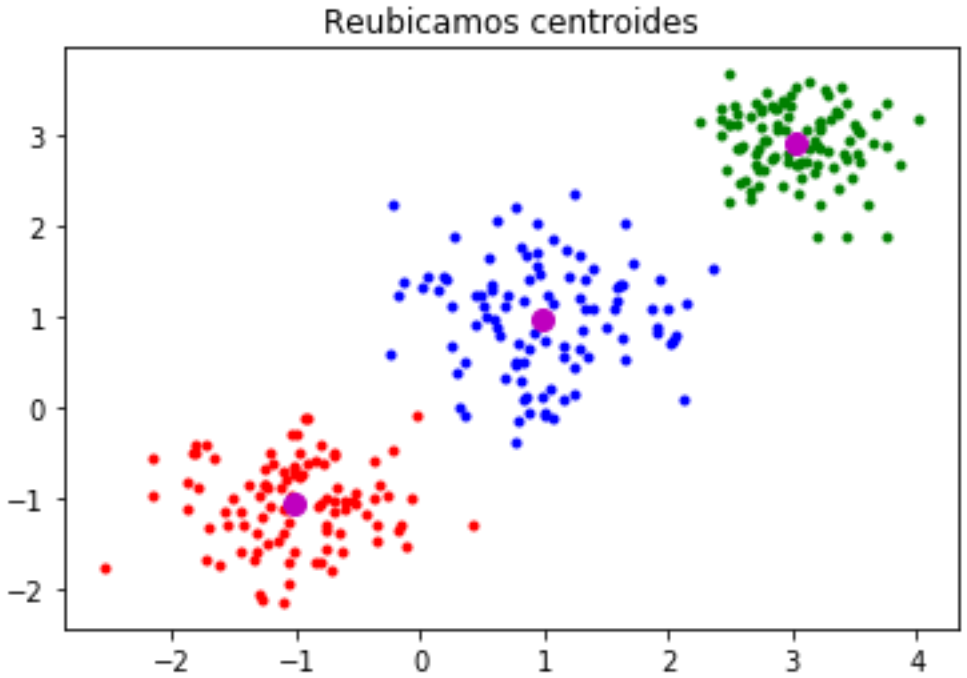


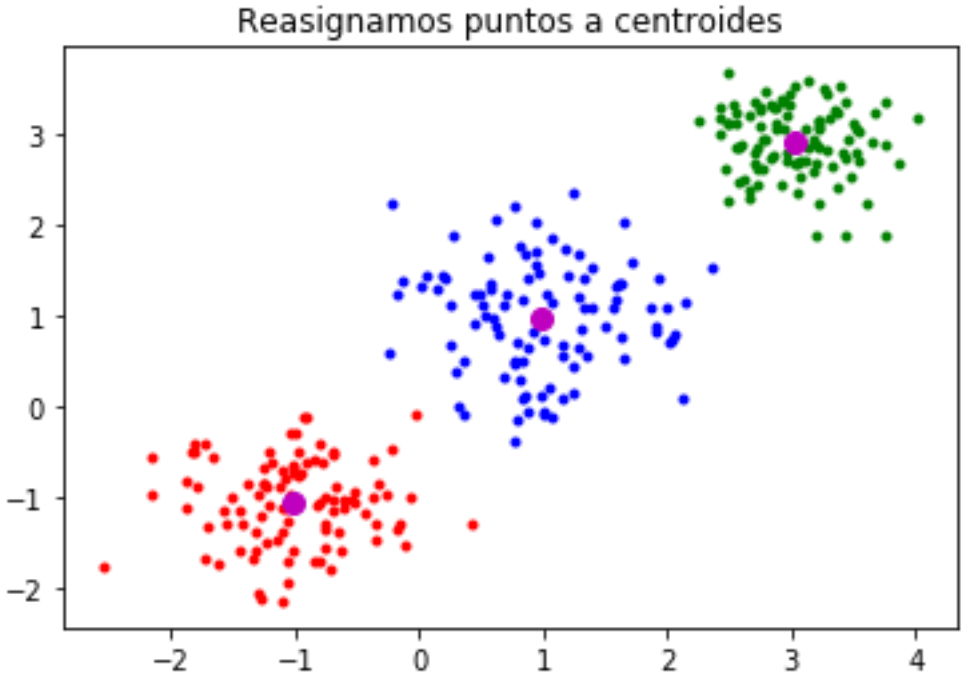
Iteración 5



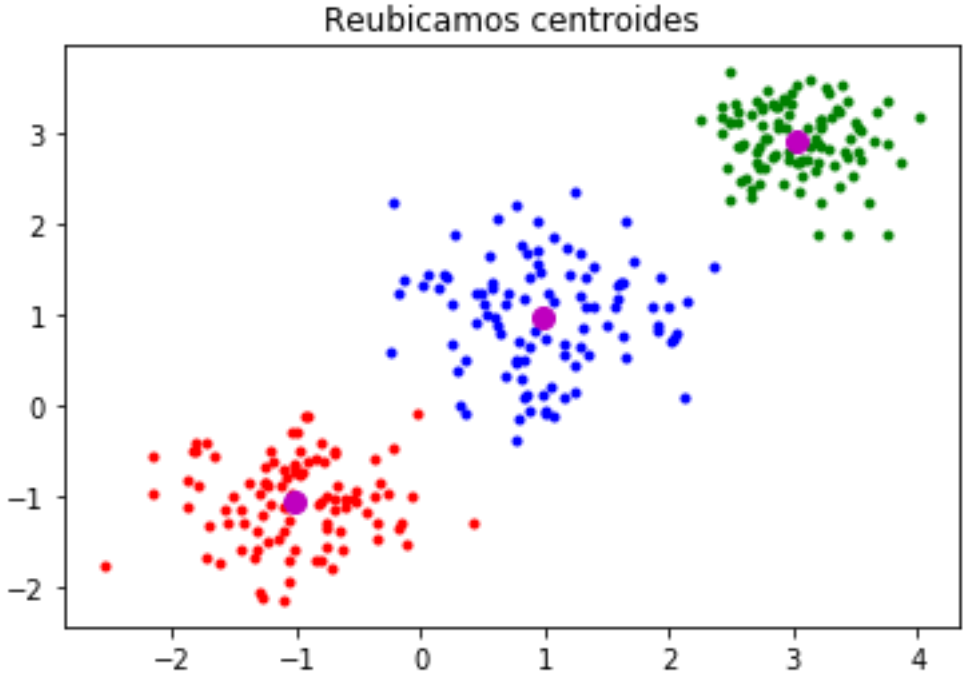

Reasignamos puntos a centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

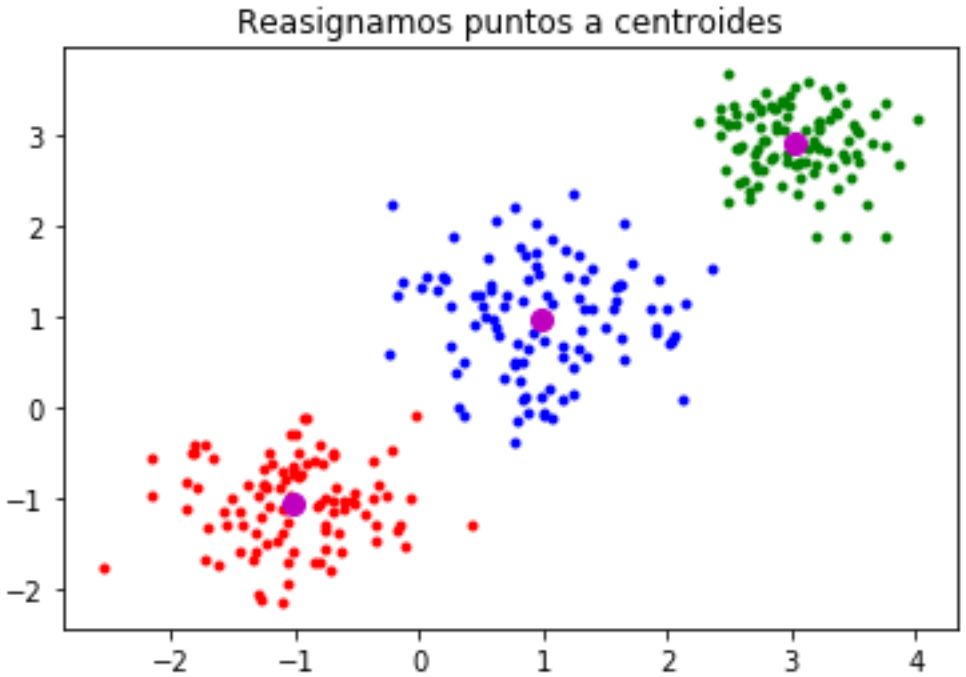
Iteración 6


Reubicamos centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

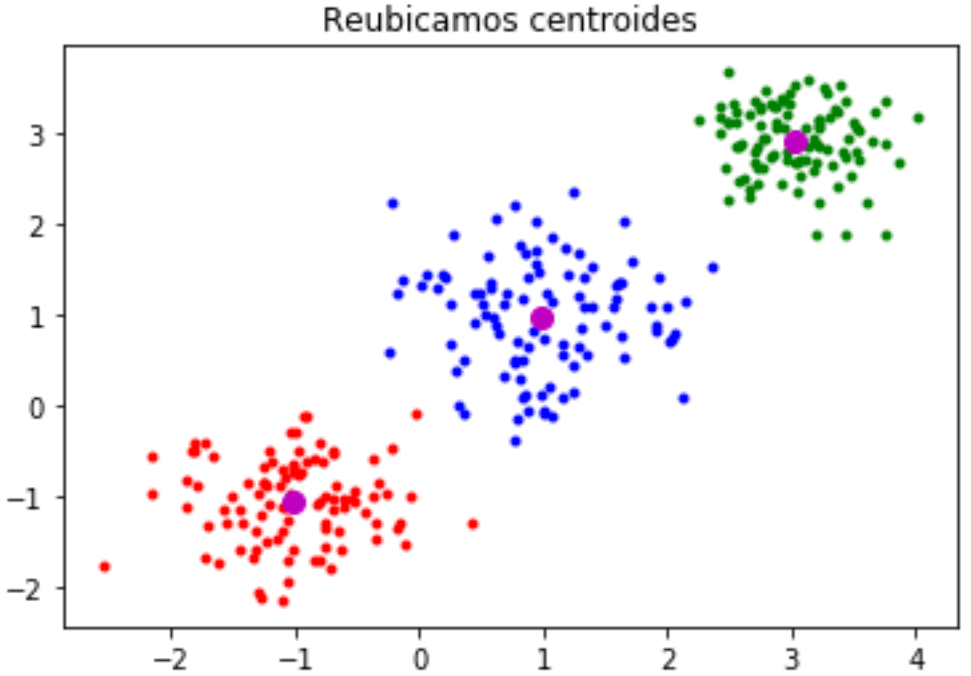

Reasignamos puntos a centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

Iteración 7


Reubicamos centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4


Reasignamos puntos a centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

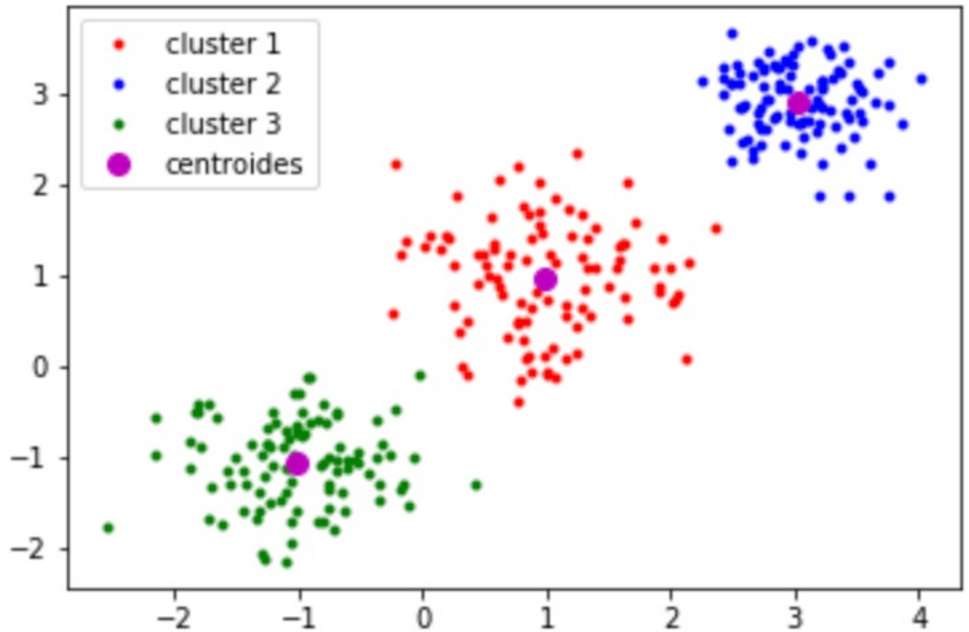
Iteración 8


Reubicamos centroides
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

```
from sklearn.cluster import KMeans
```

```
n = 3
k_means = KMeans(n_clusters=n)
k_means.fit(X)
```

```
centroides = k_means.cluster_centers_
etiquetas = k_means.labels_
```




```
plt.plot(X[etiquetas==0,0],X[etiquetas==0,1],'r.', label='cluster 1')
plt.plot(X[etiquetas==1,0],X[etiquetas==1,1],'b.', label='cluster 2')
plt.plot(X[etiquetas==2,0],X[etiquetas==2,1],'g.', label='cluster 3')

plt.plot(centroides[:,0],centroides[:,1],'mo',markersize=8, label='centroides')

plt.legend(loc='best')
plt.show()
```