taskC_DONE

October 3, 2019

[0]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
%load_ext autoreload
%autoreload 2
```

1 Data Generation

[0]:
```
from numpy.random import rand, randn
```

[0]:
```
n, d, k = 100, 2, 2
```

[0]:
```
np.random.seed(20)
X = rand(n, d)

# means = [rand(d)  for _ in range(k)]  # works for any k
means = [rand(d) * 0.5 + 0.5 , - rand(d)  * 0.5 + 0.5]  # for better plotting␣
 ↪when k = 2

S = np.diag(rand(d))

sigmas = [S]*k # we'll use the same Sigma for all clusters for better visual␣
 ↪results

print(means)
print(sigmas)
```

```
[array([0.69872366, 0.75176984]), array([0.25997411, 0.14504062])]
[array([[0.01764816, 0.        ],
       [0.        , 0.06360523]]), array([[0.01764816, 0.        ],
       [0.        , 0.06360523]])]
```

2 Solution

The log-likelihood is given by

$$\log \mathcal{L} = -\frac{1}{2}(\vec{x}_n - \vec{\mu})^T \Sigma^{-1} (\vec{x}_n - \vec{\mu}) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|)$$

The first term can be written as

$$\sum_i \sum_j -\frac{1}{2}(x_{ni} - \mu_i)(\Sigma^{-1})_{ij}(x_{nj} - \mu_j)$$

$$-\frac{1}{2}\sum_i \sum_j A_{ni}(\Sigma^{-1})_{ij}A_{nj}$$

$$-\frac{1}{2}\sum_j (A \cdot \Sigma^{-1})_{nj}A_{nj}, \qquad (A \cdot \Sigma^{-1})_{nj} = \sum_i A_{ni}(\Sigma^{-1})_{ij}$$

Where we use matrix multiplication $(\cdot)$ to take care of the first product (i-sum). However, since sum over n is not implied, that is, n is a free index, we multiply element-wise and sum over index j.

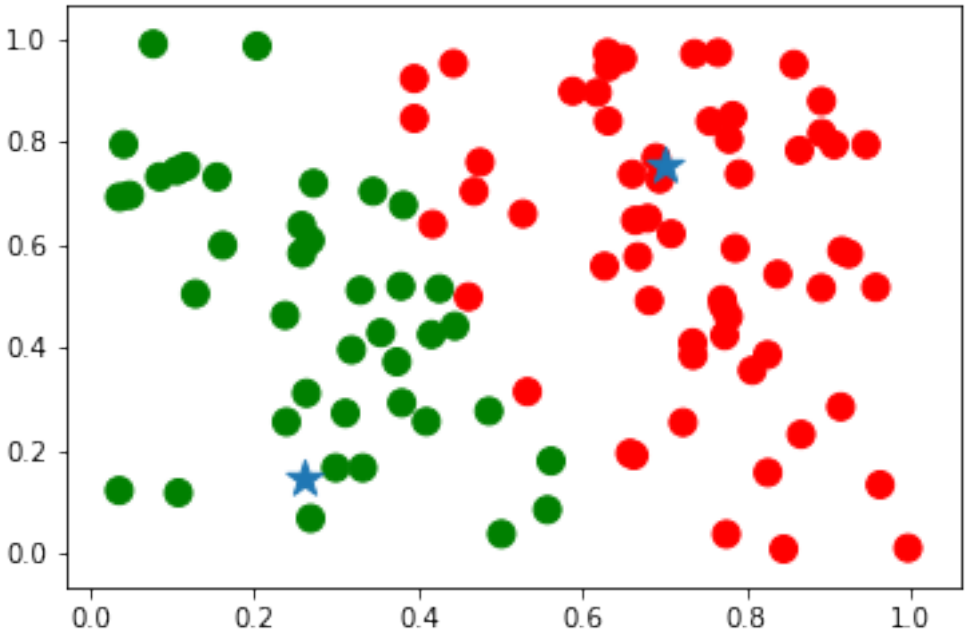
[0]:
```
def compute_log_p(X, mean, sigma):
    ''' fill your code in here...
    '''
    d = X.shape[1]
    A = X-mean
    inv_sigma = np.linalg.inv(sigma)
    prefactor = np.log(2 * np.pi)*(-d/2) * np.log(np.linalg.det(sigma))*(-1./2)
    exponent = -0.5*np.sum(A.dot(inv_sigma) * A, axis=1)
    return exponent + prefactor
```

[0]:
```
log_ps = [compute_log_p(X, m, s) for m, s in zip(means, sigmas)]  # exercise:␣
↪try to do this without looping
```

[0]:
```
assignments = np.argmax(log_ps, axis=0)
print(assignments)
```

```
[0 0 1 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 0 1 1 0 0 1 1 0 0
 1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1
 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0]
```

[0]:
```
colors = np.array(['red', 'green'])[assignments]
plt.scatter(X[:, 0], X[:, 1], c=colors, s=100)
plt.scatter(np.array(means)[:, 0], np.array(means)[:, 1], marker='*', s=200)
plt.show()
```



[0]:

[0]: