bayesian_with_python

November 26, 2021

1 2A.ml - Bayesian models with Python

Modèles de mélanges de lois. Statistiques bayésiennes. *bayespy*, *scikit-learn*.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

You can read Probabilistic Programming and Bayesian Methods for Hackers. Results might be different between examples. The example used is the same but the default parameters the optimisation uses are different.
We try different python model to deal with a Bayesian problem: a Gaussian Mixture. We will use the following example.

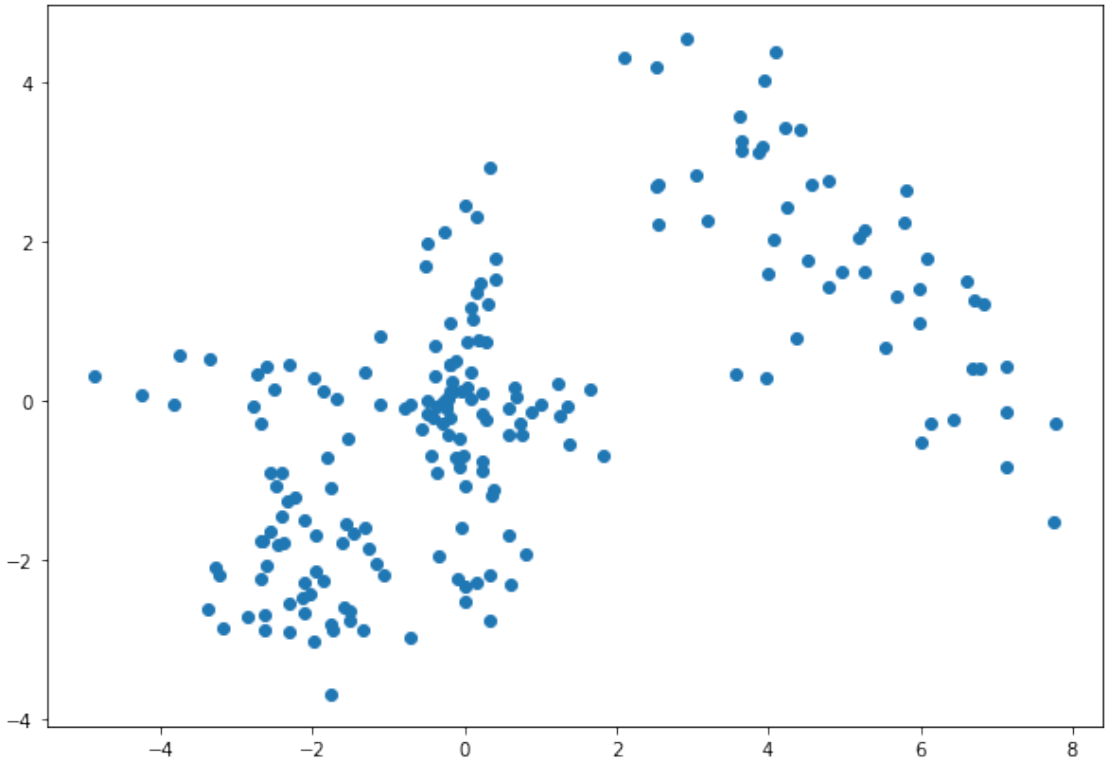
[2]:
```
%matplotlib inline
```

[3]:
```
import matplotlib.pyplot as plt
```

[4]:
```
import numpy as np
y0 = np.random.multivariate_normal([0, 0], [[2, 0], [0, 0.1]], size=50)
y1 = np.random.multivariate_normal([0, 0], [[0.1, 0], [0, 2]], size=50)
y2 = np.random.multivariate_normal([5, 2], [[2, -1.5], [-1.5, 2]], size=50)
y3 = np.random.multivariate_normal([-2, -2], [[0.5, 0], [0, 0.5]], size=50)
y = np.vstack([y0, y1, y2, y3])
X=y

fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot("111")
ax.plot(y[:,0], y[:,1], "o");
```

```
c:\python372_x64\lib\site-packages\ipykernel_launcher.py:10:
MatplotlibDeprecationWarning: Passing non-integers as three-element position
specification is deprecated since 3.3 and will be removed two minor releases
later.
  # Remove the CWD from sys.path while we load stuff.
```



bayespy

The module bayespy allows to build and estimate simple bayesian models. I just replicate the example on the Gaussian mixture model.

We define the model:

[5]:
```
from bayespy import __version__ as v1
from numpy import __version__ as v2
v1, v2
```

[5]:
```
('0.5.18', '1.18.1')
```

Si cela ne marche pas avec la version 1.14 de numpy, vous devriez essayez la version 1.13 (a priori, cette exception pose problème).

[6]:
```
N = 200  # number of data vectors
D = 2    # dimension
K = 10   # maximum number of clusters
```

[7]:
```
from bayespy.nodes import Dirichlet, Categorical, Gaussian, Wishart, Mixture
alpha = Dirichlet(1e-5*np.ones(K), name='alpha')
Z = Categorical(alpha, plates=(N,), name='z')
mu = Gaussian(np.zeros(D), 1e-5*np.identity(D), plates=(K,), name='mu')
sigma = Wishart(D, 1e-5*np.identity(D), plates=(K,), name='Lambda')
Y = Mixture(Z, Gaussian, mu, sigma, name='Y')
```

[8]:
```
Z.initialize_from_random()
```

```
c:\python372_x64\lib\site-
packages\bayespy\inference\vmp\nodes\categorical.py:43: FutureWarning: Using a
non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
  u0[[np.arange(np.size(x)), np.ravel(x)]] = 1
```

[9]:
```
from bayespy.inference import VB
Q = VB(Y, mu, sigma, Z, alpha)
```

[10]:
```
Y.observe(y)
```

[11]:
```
import time
if not hasattr(time, 'clock'):
    # bayespy still clock and it was removed in python 3.8
    setattr(time, 'clock', time.perf_counter)
```

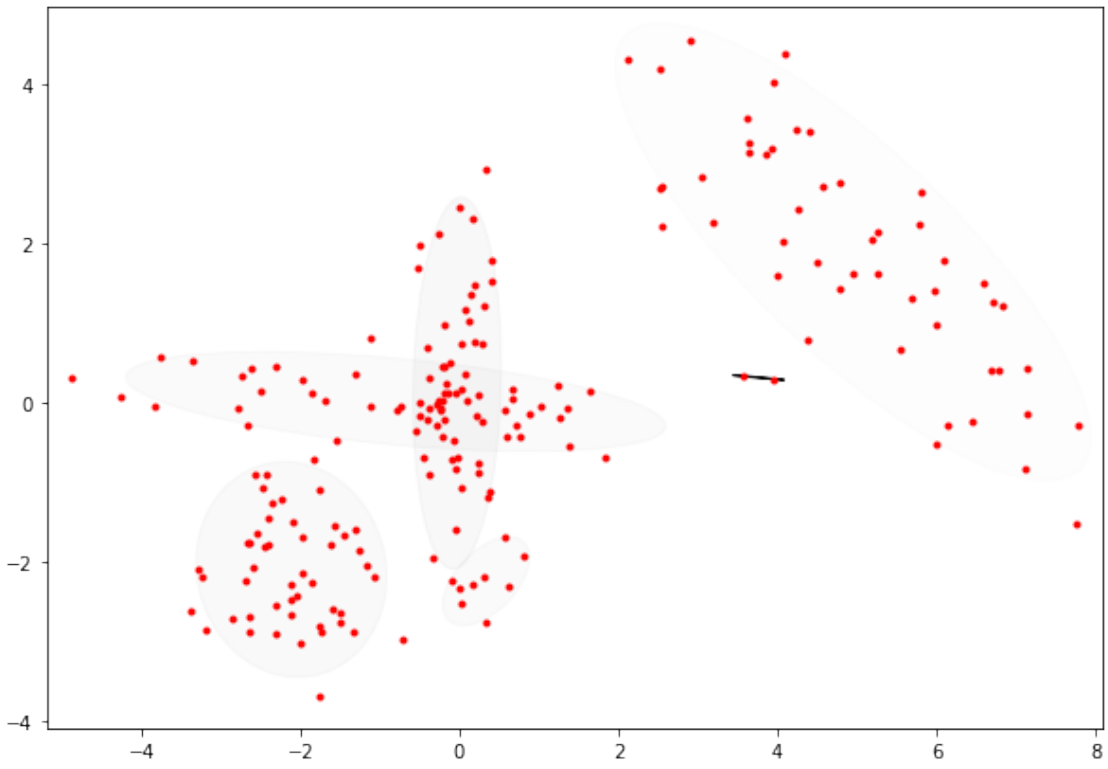
[12]:
```
Q.update(repeat=1000)
```

```
Iteration 1: loglike=-1.506353e+03 (0.011 seconds)
Iteration 2: loglike=-1.367991e+03 (0.007 seconds)
Iteration 3: loglike=-1.357210e+03 (0.006 seconds)
Iteration 4: loglike=-1.349856e+03 (0.006 seconds)
Iteration 5: loglike=-1.345145e+03 (0.007 seconds)
Iteration 6: loglike=-1.340926e+03 (0.007 seconds)
Iteration 7: loglike=-1.336195e+03 (0.008 seconds)
Iteration 8: loglike=-1.330775e+03 (0.010 seconds)
Iteration 9: loglike=-1.324233e+03 (0.009 seconds)
Iteration 10: loglike=-1.315435e+03 (0.009 seconds)
Iteration 11: loglike=-1.302308e+03 (0.009 seconds)
Iteration 12: loglike=-1.276927e+03 (0.007 seconds)
Iteration 13: loglike=-1.243768e+03 (0.011 seconds)
Iteration 14: loglike=-1.180402e+03 (0.013 seconds)
Iteration 15: loglike=-1.159467e+03 (0.010 seconds)
Iteration 16: loglike=-1.118931e+03 (0.012 seconds)
Iteration 17: loglike=-1.099640e+03 (0.010 seconds)
Iteration 18: loglike=-1.079897e+03 (0.010 seconds)
Iteration 19: loglike=-1.050285e+03 (0.012 seconds)
Iteration 20: loglike=-1.048622e+03 (0.016 seconds)
Iteration 21: loglike=-1.047742e+03 (0.015 seconds)
Iteration 22: loglike=-1.047045e+03 (0.015 seconds)
Iteration 23: loglike=-1.046335e+03 (0.014 seconds)
Iteration 24: loglike=-1.045493e+03 (0.013 seconds)
Iteration 25: loglike=-1.044402e+03 (0.007 seconds)
Iteration 26: loglike=-1.042941e+03 (0.009 seconds)
Iteration 27: loglike=-1.040920e+03 (0.007 seconds)
Iteration 28: loglike=-1.038304e+03 (0.008 seconds)
Iteration 29: loglike=-1.035139e+03 (0.007 seconds)
Iteration 30: loglike=-1.031214e+03 (0.007 seconds)
Iteration 31: loglike=-1.026811e+03 (0.007 seconds)
Iteration 32: loglike=-1.022499e+03 (0.007 seconds)
Iteration 33: loglike=-1.019083e+03 (0.007 seconds)
```

```
Iteration 34: loglike=-1.015686e+03 (0.007 seconds)
Iteration 35: loglike=-1.012240e+03 (0.007 seconds)
Iteration 36: loglike=-1.008897e+03 (0.006 seconds)
Iteration 37: loglike=-1.006014e+03 (0.006 seconds)
Iteration 38: loglike=-1.003774e+03 (0.006 seconds)
Iteration 39: loglike=-1.002331e+03 (0.006 seconds)
Iteration 40: loglike=-1.001357e+03 (0.007 seconds)
Iteration 41: loglike=-1.000439e+03 (0.006 seconds)
Iteration 42: loglike=-9.994103e+02 (0.009 seconds)
Iteration 43: loglike=-9.982072e+02 (0.013 seconds)
Iteration 44: loglike=-9.969121e+02 (0.009 seconds)
Iteration 45: loglike=-9.957966e+02 (0.007 seconds)
Iteration 46: loglike=-9.949406e+02 (0.007 seconds)
Iteration 47: loglike=-9.942194e+02 (0.008 seconds)
Iteration 48: loglike=-9.936963e+02 (0.008 seconds)
Iteration 49: loglike=-9.934065e+02 (0.007 seconds)
Iteration 50: loglike=-9.931942e+02 (0.013 seconds)
Iteration 51: loglike=-9.929650e+02 (0.010 seconds)
Iteration 52: loglike=-9.926752e+02 (0.009 seconds)
Iteration 53: loglike=-9.922659e+02 (0.011 seconds)
Iteration 54: loglike=-9.916711e+02 (0.010 seconds)
Iteration 55: loglike=-9.910285e+02 (0.008 seconds)
Iteration 56: loglike=-9.907221e+02 (0.011 seconds)
Iteration 57: loglike=-9.906515e+02 (0.009 seconds)
Iteration 58: loglike=-9.906359e+02 (0.011 seconds)
Iteration 59: loglike=-9.906304e+02 (0.010 seconds)
Converged at iteration 59.
```

[13]:
```
import bayespy.plot as bpplt
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot("111")
bpplt.gaussian_mixture_2d(Y, alpha=alpha, scale=2, color="black", fill=True, axes=ax)
```

```
c:\python372_x64\lib\site-packages\ipykernel_launcher.py:3:
MatplotlibDeprecationWarning: Passing non-integers as three-element position
specification is deprecated since 3.3 and will be removed two minor releases
later.
  This is separate from the ipykernel package so we can avoid doing imports
until
```



We get the result of the optimization:

[14]:
```
from bayespy.inference.vmp.nodes.gaussian import GaussianWishartMoments
parent = Y.parents[1]
(mu, _, sigma, _) = parent.get_moments()
mu, sigma
```

[14]:
```
(array([[ 0.00000000e+00,  0.00000000e+00],
        [ 8.57941967e+00,  8.15795231e+00],
        [ 0.00000000e+00,  0.00000000e+00],
        [ 2.80981825e+01, -4.35193132e+01],
        [ 0.00000000e+00,  0.00000000e+00],
        [ 1.87193327e+04,  1.98518969e+05],
        [-3.48238776e-01, -7.89573109e-01],
        [-6.26191082e+00, -4.95469867e+00],
        [-4.18917519e-01,  1.89822374e-01],
        [ 0.00000000e+00,  0.00000000e+00]]),
 array([[[ 2.00000000e+05,  0.00000000e+00],
         [ 0.00000000e+00,  2.00000000e+05]],

        [[ 1.30968827e+00,  1.10532747e+00],
         [ 1.10532747e+00,  1.41753118e+00]],

        [[ 2.00000000e+05,  0.00000000e+00],
         [ 0.00000000e+00,  2.00000000e+05]],

        [[ 1.87063233e+01, -9.71803333e+00],
```

```
 [-9.71803333e+00,  1.79554082e+01]],

[[ 2.00000000e+05,  0.00000000e+00],
 [ 0.00000000e+00,  2.00000000e+05]],

[[ 2.64120720e+03,  2.78178483e+04],
 [ 2.78178483e+04,  2.97301631e+05]],

[[ 4.57228969e-01,  1.21695082e+00],
 [ 1.21695082e+00,  1.34959327e+01]],

[[ 2.80059229e+00,  1.67572746e-01],
 [ 1.67572746e-01,  2.19659970e+00]],

[[ 1.31586913e+01, -2.90503809e-01],
 [-2.90503809e-01,  7.40824121e-01]],

[[ 2.00000000e+05,  0.00000000e+00],
 [ 0.00000000e+00,  2.00000000e+05]]]))
```

[15]:
```
import numpy as np
mu2 = np.linalg.solve(sigma, mu)
mu2
```

[15]:
```
array([[ 0.        ,  0.        ],
       [ 4.95354328,  1.89249089],
       [ 0.        ,  0.        ],
       [ 0.33794056, -2.24083993],
       [ 0.        ,  0.        ],
       [ 3.7638675 ,  0.31555923],
       [-0.79725482,  0.01338528],
       [-2.11059325, -2.09461048],
       [-0.02640761,  0.24587599],
       [ 0.        ,  0.        ]])
```

The way you can build your model is quite nice but it still needs some development. scikit-learn proposes a better interface.

scikit-learn

We try to solve the same problem with another module: scikit-learn.

[16]:
```
from sklearn import mixture
gmm = mixture.GaussianMixture (n_components=10, covariance_type='full')
gmm.fit(X)
```

[16]:
```
GaussianMixture(n_components=10)
```

[17]:
```
dpgmm = mixture.BayesianGaussianMixture(n_components=10, covariance_type='full')
dpgmm.fit(X)
```

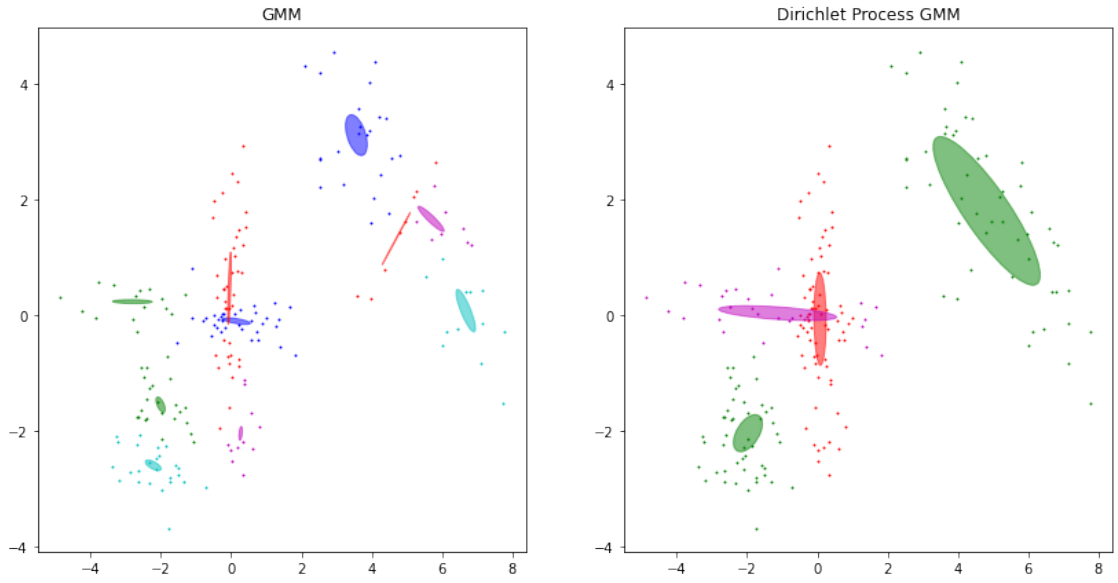
[17]:
```
BayesianGaussianMixture(n_components=10)
```

[18]:
```
import itertools
import matplotlib as mpl
color_iter = itertools.cycle(['r', 'g', 'b', 'c', 'm'])
```

```
f, axarr = plt.subplots(1, 2, figsize=(14,7))

for i, (clf, title) in enumerate([(gmm, 'GMM'),
                                  (dpgmm, 'Dirichlet Process GMM')]):
    splot = axarr[i]
    Y_ = clf.predict(X)
    for i, (mean, covar, color) in enumerate(zip(
            clf.means_, clf.covariances_, color_iter)):
        v, w = np.linalg.eigh(covar)
        u = w[0] / np.linalg.norm(w[0])
        # as the DP will not use every component it has access to
        # unless it needs it, we shouldn't plot the redundant
        # components.
        if not np.any(Y_ == i):
            continue
        splot.scatter(X[Y_ == i, 0], X[Y_ == i, 1], 0.8, color=color)

        # Plot an ellipse to show the Gaussian component
        angle = np.arctan(u[1] / u[0])
        angle = 180 * angle / np.pi  # convert to degrees
        ell = mpl.patches.Ellipse(mean, v[0], v[1], 180 + angle, color=color)
        ell.set_clip_box(splot.bbox)
        ell.set_alpha(0.5)
        splot.add_artist(ell)
        splot.set_title(title)
```



[19]: