[Lab] Gaussian Mixture Model (GMM)

Jae Yun JUN KIM*

August 20, 2019

1 Gaussian Mixture Models

Sources: Scikit-learn

1.1 Example 1: Density estimation for a Gaussian mixture

Plot the density estimation of a mixture of two Gaussians. Data is generated from two Gaussians with different centers and covariance matrices.

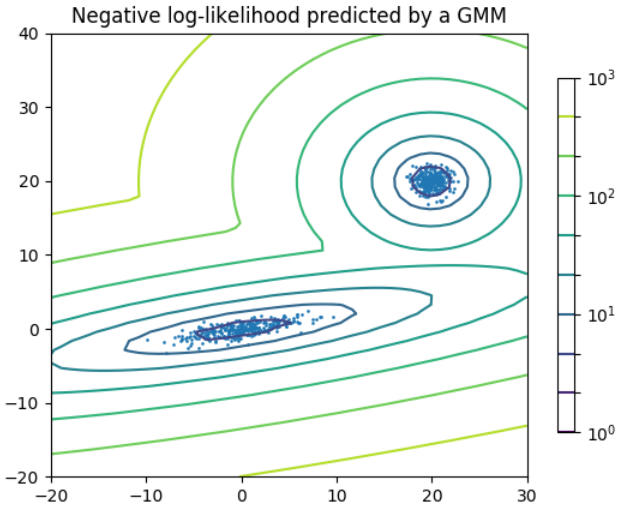

Figure 1: Density estimation

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from sklearn import mixture

n_samples = 300

# generate random sample, two components
np.random.seed(0)

# generate spherical data centered on (20, 20)
```

*ECE Paris Graduate School of Engineering, 37 quai de Grenelle 75015 Paris, France; jae-yun.jun-kim@ece.fr

```
shifted_gaussian = np.random.randn(n_samples, 2) + np.array([20, 20])

# generate zero centered stretched Gaussian data
C = np.array([[0., -0.7], [3.5, .7]])
stretched_gaussian = np.dot(np.random.randn(n_samples, 2), C)

# concatenate the two datasets into the final training set
X_train = np.vstack([shifted_gaussian, stretched_gaussian])

# fit a Gaussian Mixture Model with two components
clf = mixture.GaussianMixture(n_components=2, covariance_type='full')
clf.fit(X_train)

# display predicted scores by the model as a contour plot
x = np.linspace(-20., 30.)
y = np.linspace(-20., 40.)
X, Y = np.meshgrid(x, y)
XX = np.array([X.ravel(), Y.ravel()]).T
Z = -clf.score_samples(XX)
Z = Z.reshape(X.shape)

CS = plt.contour(X, Y, Z, norm=LogNorm(vmin=1.0, vmax=1000.0),
                 levels=np.logspace(0, 3, 10))
CB = plt.colorbar(CS, shrink=0.8, extend='both')
plt.scatter(X_train[:, 0], X_train[:, 1], .8)

plt.title('Negative log-likelihood predicted by a GMM')
plt.axis('tight')
plt.show()
```

1.2 Example 2: Voice gender detection using GMMs

Source: https://appliedmachinelearning.wordpress.com/2017/06/14/voice-gender-detection-using-gmms-a-python-primer/
Download the dataset file from the `data` folder available in:
https://www.dropbox.com/sh/c17hegex6zfe73c/AAD17hty-a2YC3alByicd1YCa?dl=0
Script for training the GMM:

```
import os
import _pickle as cPickle # for python 3.x
# import cPickle # for python 2.x
import numpy as np
from scipy.io.wavfile import read
from sklearn.mixture import GMM
import python_speech_features as mfcc
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")

def get_MFCC(sr,audio):
    features = mfcc.mfcc(audio,sr, 0.025, 0.01, 13,appendEnergy = False)
    features = preprocessing.scale(features)
    return features

#path to training data
source   = "pygender\\train_data\\youtube\\female\\"
# source   = "pygender\\train_data\\youtube\\male\\"
#path to save trained model
dest     = "pygender\\"
files    = [os.path.join(source,f) for f in os.listdir(source) if
            f.endswith('.wav')]
features = np.asarray(());

for f in files:
    sr,audio = read(f)
    vector   = get_MFCC(sr,audio)
    if features.size == 0:
        features = vector
    else:
        features = np.vstack((features, vector))

gmm = GMM(n_components = 8, n_iter = 200, covariance_type='diag',
        n_init = 3)
gmm.fit(features)
picklefile = f.split("\\")[-2].split(".wav")[0]+".gmm"

# model saved as male.gmm
#cPickle.dump(gmm, open(dest + picklefile, 'w'))
cPickle.dump(gmm,open(dest + picklefile,'wb'))
print('modeling completed for gender:',picklefile)
```

Script for testing the GMM:

```
import os
import _pickle as cPickle # for python 3.x
# import cPickle # for python 2.x
import numpy as np
from scipy.io.wavfile import read
import python_speech_features as mfcc
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")
def get_MFCC(sr,audio):
    features = mfcc.mfcc(audio,sr, 0.025, 0.01, 13,appendEnergy = False)
    feat     = np.asarray(())
    for i in range(features.shape[0]):
        temp = features[i,:]
        if np.isnan(np.min(temp)):
            continue
        else:
            if feat.size == 0:
                feat = temp
            else:
                feat = np.vstack((feat, temp))
    features = feat;
    features = preprocessing.scale(features)
    return features

#path to test data
sourcepath = "pygender\\test_data\\AudioSet\\female_clips\\"
#path to saved models
modelpath  = "pygender\\"

gmm_files = [os.path.join(modelpath,fname) for fname in
              os.listdir(modelpath) if fname.endswith('.gmm')]
models    = [cPickle.load(open(fname,'rb')) for fname in gmm_files]
genders   = [fname.split("\\")[-1].split(".gmm")[0] for fname
              in gmm_files]
files     = [os.path.join(sourcepath,f) for f in os.listdir(sourcepath)
              if f.endswith(".wav")]

for f in files:
    print(f.split("\\")[-1])
    sr, audio  = read(f)
    features   = get_MFCC(sr,audio)
    scores     = None
    log_likelihood = np.zeros(len(models))
    for i in range(len(models)):
        gmm    = models[i]         #checking with each model one by one
```

```
        scores = np.array(gmm.score(features))
        log_likelihood[i] = scores.sum()
    winner = np.argmax(log_likelihood)
    print("\tdetected as - ", genders[winner],"\n\tscores:female ",
     log_likelihood[0],",male ", log_likelihood[1],"\n")
```