Generative Models 2: MLE, MAP and EM

Virtual Class

Dr. Jean Marc Odobez (https://www.idiap.ch/~odobez)
Samy Tafasca (https://www.idiap.ch/~stafasca)

2021-2022

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)

1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

Overview

Main points

- Submissions were great overall - well done!
- GMMs are a class of generative models, so we can use them to synthesize new data samples
- EM is a powerful algorithm to train GMMs for clustering or density estimation tasks
- K-means is a special case of GMMs which assumes that clusters have spherical covariance matrices, and that samples are distributed uniformly across clusters
- Initialization is important for EM. It can be set randomly, using K-means centroids or prior knowledge
- The EM can be extended to MAP estimation instead of ML estimation in order to benefit from prior distributions and the bayesian framework.

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)

1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

GMM as a Generative Model

Exercice 1

Complete the `sample` function below which should use ancestral sampling to generate new data points from a given gaussian mixture.

Solution

```python
def sample(N, pis, mus, sigmas):
    # Number of Mixtures
    K=len(pis)

    # Initialize x and z
    x=np.empty((N, 2))
    z=np.empty(N, dtype=np.int)

    ### BEGIN SOLUTION
    for n in range(N):
        mixture=int(np.random.choice(K, size=1, p=pis)[0])
        sample=mvn.rvs(mean=mus[mixture], cov=sigmas[mixture])
        x[n]=sample
        z[n]=mixture
    ### END SOLUTION

    return x, z
```

GMM as a Generative Model

Exercice 2

Using the `sample` function, complete the following function to generate a sample of $N = 400$ synthetic data points from a 2-dimensional GMM with $K = 3$ components, with known prior, mean, and diagonal covariance matrix. Assume $\pi_1 = 0.25$, $\pi_2 = 0.5$, $\pi_3 = 0.25$, $\mu_1 = (-1, 0)$, $\mu_2 = (1, 1)$, $\mu_3 = (2, 2)$ and $\Sigma_1 = \Sigma_2 = \Sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Solution

```
def sampleMyData(N):

    ### BEGIN SOLUTION
    K =3
    pis =np.array([0.25, 0.5, 0.25])
    mus =np.array([[-1, 0], [1., 1.], [2., 2.]])
    sigmas =np.tile([[1., 0.], [0., 1.]], (3, 1, 1))

    x, z =sample(N, pis, mus, sigmas)
    ### END SOLUTION

    return x, z, pis, mus, sigmas
```

GMM as a Generative Model

Exercice 3

Complete the function `SkLearnGMMfitting` below. Use the `GaussianMixture` class from the module `sklearn.mixture` to train a GMM on the synthetic data (which was imported at the beginning of the lab). Initialize the model with $K = 3$ components, mean vector μ using K-means, the mixing coefficients π with uniform values (i.e. equal probability for each mixture) and the covariance with $\Sigma_1 = \Sigma_2 = \Sigma_3 = \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$

Solution

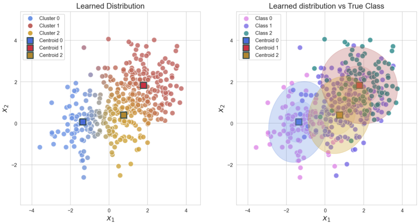
All 4 required parts are grouped together in the function below.

```
def SkLearnGMMfitting(X, K):
    ### BEGIN SOLUTION
    pis =np.ones(K) /K
    sigmas =np.tile([[0.25, 0.], [0., 0.25]], (K, 1, 1))
    gmm =GaussianMixture(n_components =K, init_params ="kmeans", weights_init =pis,
                                        precisions_init =np.linalg.inv(sigmas))
    gmm.fit(X)
    gammas =gmm.predict_proba(X)
    piest =gmm.weights_
    muest =gmm.means_
    sigmaest =gmm.covariances_
    ### END SOLUTION
    return piest, muest, sigmaest, gammas
```

GMM as a Generative Model

Question 1

Compare the true parameters with the estimated parameters. What do you observe? (comment on the mixture parameters, the mean, the covariances)



	π_1	π_2	π_3	μ_1	μ_2	μ_3	Σ_1
True	0.25	0.50	0.25	(-1.0, 0.0)	(1.0, 1.0)	(2.0, 2.0)	$\begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$
Pred	0.23	0.37	0.41	(-1.4, 0.1)	(0.8, 0.4)	(1.8, 1.8)	$\begin{pmatrix} 0.62 & 0.13 \\ 0.13 & 0.95 \end{pmatrix}$

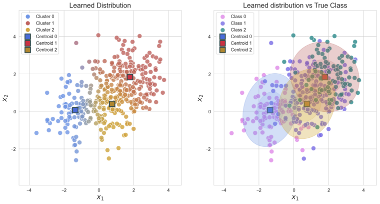
Solution

- Mixing coefficients are somewhat different, especially for class 1 and 2.
- Means and covariance matrices are sometimes close, but other times different.
- One reason for this mismatch is that the true mixtures are too close to one another.

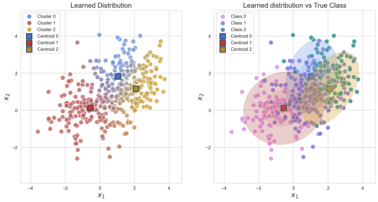
GMM as a Generative Model

Question 2

When you run the fitting several times (up to 7 times), can you observe the variability (observe the graph)? Where does this come from? Which gaussians from the original data are more affected by this variability? and why?



Run 1



Run 2

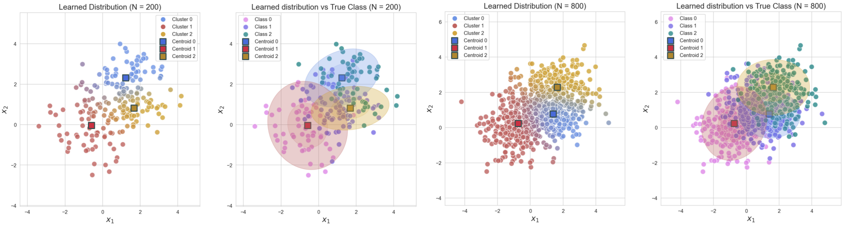
Solution

- Variability comes from initialization conditions. Since the initial π and Σ are fixed, the randomness stems mainly from the initial values of μ which are estimated from k-means.
- The gaussians that are most impacted are those corresponding to class 1 and 2, since these classes overlap the most.

GMM as a Generative Model

Question 3

Running the code, but sampling $N = 200$ data points instead of 400, what would you expect when fitting the resulting data? and vice-versa, when $N = 800$?



Solution

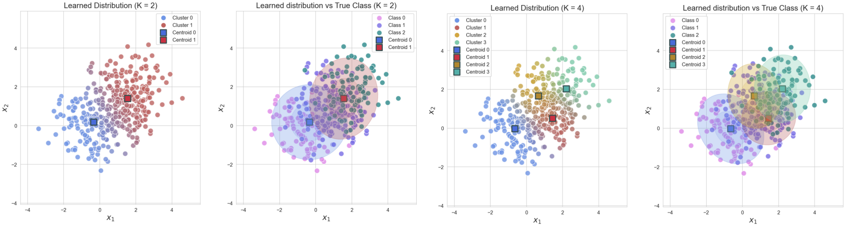
Given that the data actually follow a GMM model:

- Less data is likely going to lead to a worse model (more variability in estimates, including across runs)
- Whereas more data should in theory produce better models

GMM as a Generative Model

Question 4

Running the code with $K = 2$ and $K = 4$, what do you think about the results ?



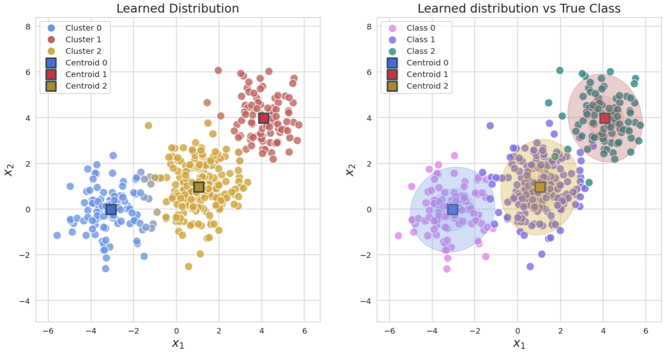
Solution

- When $K = 4$, cluster 0 is well captured (more isolated from the rest), whereas 3 mixtures are split between the 2 other overlapping clusters.
- Irrespective of the true number of mixtures that generated the observed data, a GMM trained with any value of K will still produce a model. The right value of K depends on the task, the metrics and the interpretation.

GMM as a Generative Model

Question 5

Running the code with $K = 3$, sampling $N = 400$ data points, but using as means $\mu_1 = (-3, 0)$, $\mu_2 = (1, 1)$, $\mu_3 = (4, 4)$, what can you observe in the subsequent fitting?



Solution

- In this setting, the true gaussians are better separated, so the training leads to more accurate estimation of the parameters and more stable results across runs.

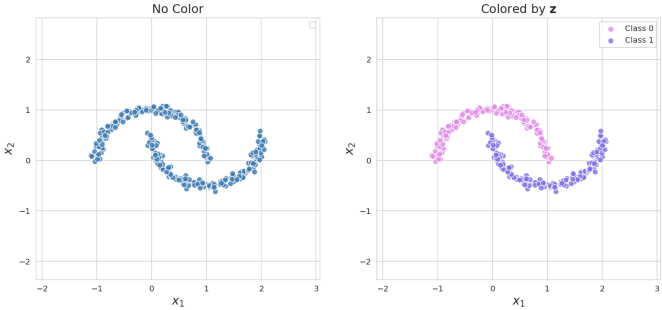
GMM as a Generative Model

Goodfellow et al. 2016, p. 65 states:

> *A Gaussian mixture model is a universal approximator of densities, in the sense that any smooth density can be approximated with any specific nonzero amount of error by a Gaussian mixture model with enough components.*

Exercice 4

Train a GMM on this synthetic data using $K = 2$ components and default values for the rest of the parameters. Then use the 'sample' function from above (Exercise 1) to generate $N = 400$ data points from the learned distribution and visualize them.



GMM as a Generative Model

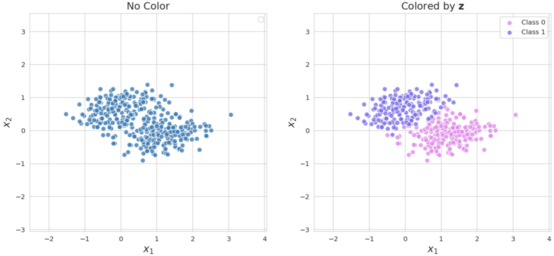
Solution

```
### BEGIN SOLUTION
K =2
gmm =GaussianMixture(n_components =K)
gmm.fit(Xmoon)
N =400
X_gen, Z_gen =sample(N, gmm.weights_, gmm.means_, gmm.covariances_)
### END SOLUTION
```

GMM as a Generative Model

Question 6

Comment on the results when $K = 2$.



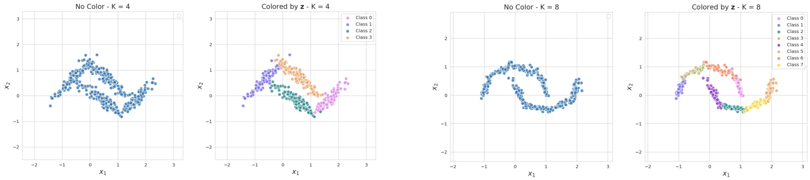
Solution

- Even if the true number of clusters in the data is 2, a GMM with 2 mixtures is not able to learn this distribution simply because the true clusters have the shape of a moon, which is very different from the ellipsoid shape of a gaussian.

GMM as a Generative Model

Question 7

Repeat the same experiment, but this time with $K = 4, 6, 8$. Comment on the result again.



Solution

- When we increase the value of K, we observe that the GMM gets better at approximating the distribution.
- It does so by approximating each portion of each moon by a gaussian.
- This is akin to approximating the curve of a non-linear function with multiple linear ones, each of them restricted to a local region.

GMM as a Generative Model

Question 8

- **a** What will happen as K increases (think of extreme cases where $K = N$)? will the fit be better? which general machine learning problem will arise?
- **b** Thinking of the GMM learning as a typical data fitting problem (here optimizing the data likelihood), and the value of K as a hyper-parameter, how could you organize your data to decide on a good value of K?

Solution

- **a**
  - As we increase K to very high values, the model becomes very flexible (i.e. complex), to the point where it can represent the training data perfectly.
  - The model captures not only the statistical regularities in the data, but also the noise that may be present.
  - This is generally called over-fitting.
- **b** There are many ways to choose a good value of K
  - Split the data into train and test partitions, then use the train partition to fit the model and evaluate the log-likelihood. Then use the test partition to evaluate the log-likelihood again and compare the two values. If the gap is large, then we are likely overfitting.
  - Use cluster performance evaluation metrics such as the Silhouette score.

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)

1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

Implementation of EM for GMMs

Exercice 5

Complete the `GMM_EStep` function.

Solution

```
def GMM_EStep(X, pis, mus, sigmas):

    N, D =X.shape
    K =mus.shape[0]

    gammas =np.zeros((N, K))

    ### BEGIN SOLUTION
    for n in range(N):
        for k in range(K):
            gammas[n, k] =(pis[k] *mvn.pdf(X[n], mean =mus[k], cov =sigmas[k],
                                            allow_singular=True))
    # Normalization
    gammas /=gammas.sum(axis =1).reshape(-1, 1)
    ### END SOLUTION

    return gammas
```

Implementation of EM for GMMs

Exercice 6

Complete the `GMM_MStep` function.

Solution

```
def GMM_MStep(X, gammas):
    N, D =X.shape
    K =gammas.shape[1]

    mus =np.zeros((K, D))
    sigmas =np.zeros((K, D, D))
    pis =np.zeros((K,))

    ### BEGIN SOLUTION
    for k in range(K):
        mus[k] =(X *gammas[:, k].reshape(-1, 1)).sum(axis =0)
    mus /=gammas.sum(axis =0).reshape(-1, 1)

    for k in range(K):
        # /!\ ddof = 1 by default which normalizes by N-1 instead of N
        sigmas[k] =np.cov(X, rowvar =False, aweights =gammas[:, k], ddof =0)

    pis =gammas.sum(axis =0) /N
    ### END SOLUTION

    return pis, mus, sigmas
```

Implementation of EM for GMMs

Exercice 8

Complete the `GMM_EM` function.

Solution

```
def GMM_EM(X, K, mus =None, sigmas =None, pis =None, kmeans_init =False, max_iter =50, tol
                                                    =0.1, verbose =True):
    # Initialization of the parameters (pi, mu, sigma)
    ...
    # Training
    ...
    # Loop
    for iteration in range(1, max_iter +1):
        for step in ('e', 'm'):
            ### BEGIN SOLUTION
            if step =='e':
                # E-Step
                gammas =GMM_EStep(X, pis, mus, sigmas)
            else:
                # M-Step
                pis, mus, sigmas =GMM_MStep(X, gammas)
            ### END SOLUTION
            # Compute Lower Bound and Evaluate Log-likelihood
            ...
        # Break if log-likelihood change is lower than tol
        ...
    return pis, mus, sigmas, history
```

Implementation of EM for GMMs

Bishop 2016, p. 434 states

> *Before discussing how to maximize this function [data log-likelihood], it is worth emphasizing that there is a significant problem associated with the maximum likelihood framework applied to Gaussian mixture models, due to the presence of singularities. This will occur whenever one of the Gaussian components 'collapses' onto a specific data point.*

Question 9

Read the documentation of scikit-learn's `GaussianMixture` class and explain how the library handles the singularity problem.

Solution

- Sklearn's `GaussianMixture` class handles the singularity problem by adding a non-negative regularization value to the diagonal of the covariance, which ensures that the covariance matrices are all positive. This is controlled by the `reg_covar` argument.

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

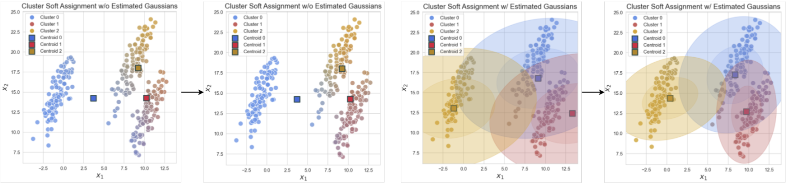
Part III: Modeling the Visual Focus Of Attention (VFOA)

1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

Testing EM for GMMs on Synthetic Data

Question 10

Visually speaking, what does the E-step do ? What about the M-step ? Does this behavior correspond to their definition?



E-step **M-step**

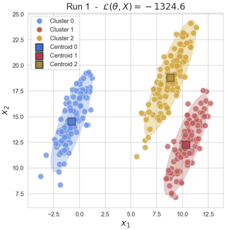
Solution

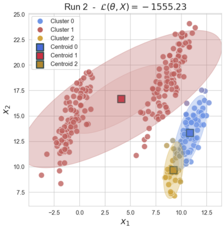
- The E-Step changes the colors of the data points, which encode cluster assignment. This correponds to computing new values for the responsibilities γ.
- The M-Step changes the location and covariance matrix of the mixtures. It also updates the mixing coefficients π, but this is difficult to see visually.
- The visual interpretations seem to be aligned with the definitions of these steps.

Testing EM for GMMs on Synthetic Data

Question 11

What is the objective function that the EM tries to optimize ? From the experiments above (run the algorithm several times), do you think the EM algorithm always converges to the global optimum of that objective function ? Justify your answer.





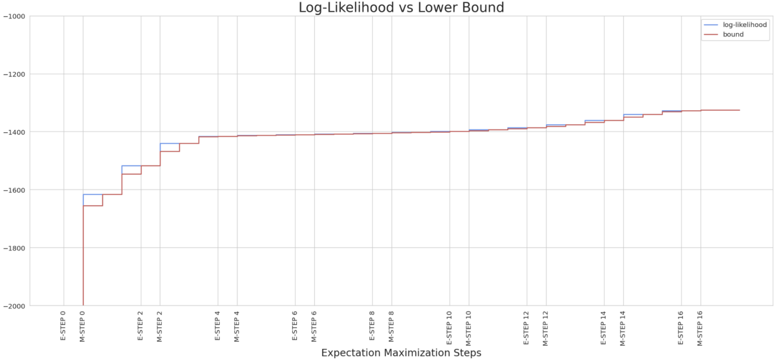
Solution

- The EM for GMM tries to (locally) maximize the incomplete data (i.e. X) log-likelihood, by maximizing its lower bound defined as, $\mathcal{L}(q, \theta) = \sum_{Z} q(Z) \ln \left\{ \frac{p(X, Z|\theta)}{q(Z)} \right\}$.
- Maximizing the lower bound during the M-step corresponds to maximizing the expected value of the complete data (i.e. X, Z) log-likelihood, under the posterior $P(Z|X, \theta)$.
- The EM has no convergence guarantee in general, and can get stuck in local optimums. In practice, this can be seen by comparing the results of different runs.

Testing EM for GMMs on Synthetic Data

Question 12

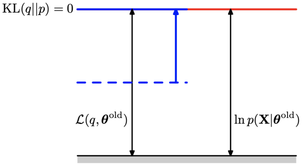
Does the behavior of the two curves match what is describe in the course with respect to the EM steps (section, why does EM work ?). What is the quantity that makes up the difference between the log-likelihood and the lower bound after each M-step?



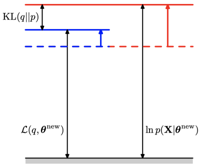
Testing EM for GMMs on Synthetic Data

Solution

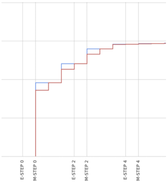
- The difference between the log-likelihood and the lower bound in general corresponds to the KL divergence between the distribution $q(Z)$ defined over the latent variables, and the posterior distribution $p(Z|X,\theta)$ over latent variables, given the parameters. After the M-step, this KL divergence represents the difference between $p(Z|X,\theta_{old})$ and $p(Z|X,\theta_{new})$.
- After each E-step, the lower bound increases to match the log-likelihood (i.e. θ remains fixed, but q changes) $\Rightarrow$ KL divergence becomes 0. This can be seen on the plotted curve.
- After each M-step, the lower bound increases, but the log-likelihood increases even more (i.e. q remains fixed, but θ changes). The difference is a new non-negative KL divergence.



E-Step



M-Step



Evolution curve

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

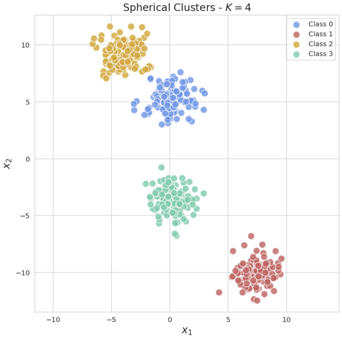
Part III: Modeling the Visual Focus Of Attention (VFOA)

1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

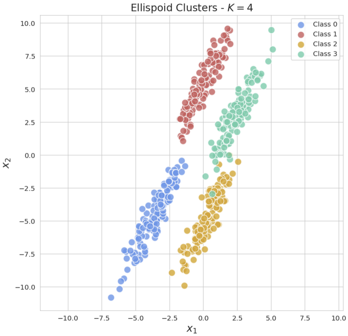
Comparing GMMs and K-means

This section is about comparing GMMs and K-means on two different datasets to illustrate their behaviors and shortcomings:

- Spherical Clusters
- Ellipsoid Clusters



Spherical Data



Ellipsoid Data

Comparing GMMs and K-means

Exercice 9

Train a GMM on the spherical dataset using the right number of components $K = 4$ and k-means initialization.

Solution

```
### BEGIN SOLUTION
K =4
pis, mus, sigmas, history =GMM_EM(X_sphere, K, max_iter =50, tol =0.1, kmeans_init =True)
### END SOLUTION
```

Exercice 10

Now train a GMM on the ellipsoid dataset using the right number of components $K = 4$ and k-means initialization.

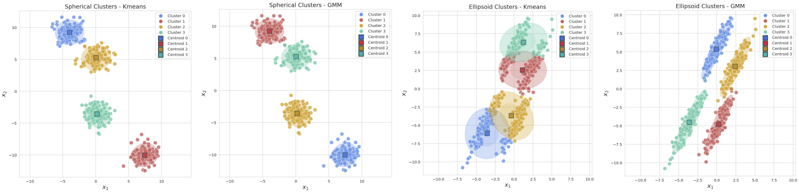
Solution

```
### BEGIN SOLUTION
K =4
pis, mus, sigmas, history =GMM_EM(X_ellipse, K, max_iter =50, tol =0.1, kmeans_init =True)
### END SOLUTION
```

Comparing GMMs and K-means

Question 13

Explain why both algorithms work similarly well on spherical data but only GMM works well on the ellipsoid data.



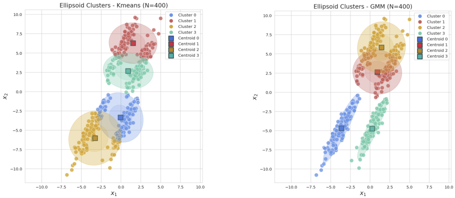
Solution

- K-means assumes isotropic clusters, whereas GMMs support gaussians with flexible covariance matrices which can stretch in any direction.
- This explains why a GMM would work on both spherical and elongated clusters when K-means would only work well on spherical ones.

Comparing GMMs and K-means

Question 14

Regenerate the ellipsoid data using only $N = 400$ data points. Re-apply the Kmeans clustering and GMM fitting on this data. What do you observe? According to you, where does this come from?



Solution

- With $N = 400$: both algorithms fail on ellipsoid data. The GMM in particular fails on two clusters.
- This is likely happening because the smaller amount of data combined with the k-means initialization leads the EM to get stuck in a bad local optima. Turning off the k-means initialization can help overcome this problem.

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)

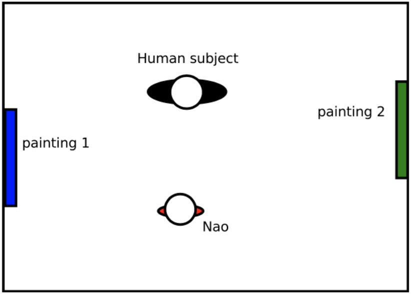
1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

Problem Description

The goal of this 3rd part is to apply GMMs to recognize the VFOA of a person, given their head orientation. In particular, we explore both supervised and unsupervised approaches in modeling the distribution of observed data points.

Context

Assume that a scene consists of a room with the Nao robot, a human and 2 paintings. Nao is interacting with the human, commenting about the paintings and having a Q&A session. It is assumed that there are 3 possible VFOA targets for the human: Nao and the two paintings. The situation is illustrated in the following figure.



Problem Description

From the video that looks at the subject, we can extract the person's head orientation $\mathrm{x} = (x_1, x_2)$, with x_1 representing the pan angle (i.e. looking right or left), and x_2 the tilt angle (i.e. looking up or down). From this observation, we would like to know whether the person looks at the robot, at the first painting, or at the second painting.

Model assumptions:

- probability of looking at a focus z is modeled according to a categorical distribution parameterized by π:

  $$p(\mathrm{z} = k|\pi) = \pi_k$$

  $k \in \{0, 1, 2\}$ ($k = 1$ - looking at painting 1, $k = 2$, painting 2, and $k = 0$, looking at Nao.

- Given z, the probability of observing a head pose is given by:

  $$p(\mathrm{x}|\mathrm{z} = k) = \mathcal{N}(\mathrm{x}|\mu_k, \sigma_k)$$

  where μ_k denotes the mean pose when looking at target k, and σ_k represents the variability associated with target k.

According to this hypothesis, the probability of observing a head pose is given by:

$$p(\mathrm{x}) = \sum_k \pi_k \mathcal{N}(\mathrm{x}|\mu_k, \sigma_k)$$

In other words, we have a Gaussian Mixture Model (GMM).

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)

1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

Supervised Learning Approach

Exercice 1

Complete the function `learn_parameters_observable` below.

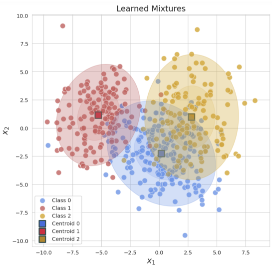
Solution

```
def learn_parameters_observable(X, Z):
    N, D =X.shape
    K =len(np.unique(Z))

    pis =np.empty(K)
    mus =np.empty((K, D))
    sigmas =np.empty((K, D, D))

    ### BEGIN SOLUTION
    for k in range(K):
        mask =(Z ==k)
        pis[k] =mask.sum() /N
        mus[k] =X[mask, :].mean(axis =0)
        sigmas[k] =np.cov(X[mask, :], rowvar =False, ddof =0)
    ### END SOLUTION

    return pis, mus, sigmas

# Learning
pis_train, mus_train, sigmas_train =learn_parameters_observable(Xtrain, Ztrain)
```

Supervised Learning Approach

Question 1

Looking at the estimated gaussians, do their locations (i.e. means) correspond to what you can expect in terms of head pose w.r.t the targets (1 corresponds to 5°) ? A person looking to their right corresponds to a positive or negative pan angle?



Solution

- Yes. An average pan of -25° (i.e. 25° to the right of the person) and an average tilt of 5° (i.e. 5° upwards) for the red cluster, seems reasonable for the average person looking at Painting 1.
  Remember **that the gaze angle is not the head angle** (usually, the head is turned half of the needed gaze angle). The same reasoning can be applied to the remaining clusters.

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)

1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)

1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

Unsupervised Learning using EM

Exercice 4

Complete the `learn_parameters_unsupervised` function using the `GMM_EM` function. Remember that we do not use Z_1 and Z_2 here, and that these will only serve to evaluate our models.

Note: Since the learning is completely unsupervised, cluster numbers don't have a meaning. In other words, the same set of points can be assigned the label 0, 1 or 2, depending on the initialization of the algorithm. We can `reorder` the cluster labels to match their respective classes.

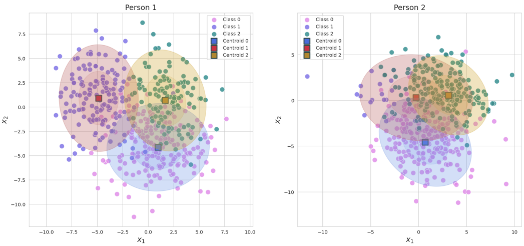
Solution

```
def learn_parameters_unsupervised(X, K, pis =None, mus =None, sigmas =None, max_iter =200,
                                  tol =0.001, kmeans_init =True):

    ### BEGIN SOLUTION
    pis, mus, sigmas =GMM_EM(X, K, pis =pis, mus =mus, sigmas =sigmas, max_iter =max_iter,
                             tol =tol, kmeans_init =kmeans_init)
    ### END SOLUTION

    pis, mus, sigmas =utils.reorder(pis, mus, sigmas, mus_train)
```

Unsupervised Learning using EM

Question 3

Do you think that the learned gaussian means match what should be expected from a semantic viewpoint? Explain your answer, noting how the interaction with the robot can (negatively) affect parameter estimation.



Solution

- The learned gaussians are mostly where we would expect them, except for Class 1 of Person 2 for which the gaussian doesn't seem to represent the cluster well. This is due to the lack of data in that class.
- Depending on their interaction with Nao, a person may be focusing more on certain targets compared to others, which can lead to a class imbalance issues.

Unsupervised Learning using EM

Question 4

Compare the performance of this unsupervised approach to what was obtained using supervised learning. Explain these results.

Solution

We obtain:

- Accuracy of Person 1: 85.98%
- Accuracy of Person 2: 77.64%

- The performance of Person 1 in the supervised case ($\sim 82\%$) is lower than the unsupervised case ($\sim 86\%$). This can be explained by the fact that the data of person 1 has a decent separation between the gaussians, so unsupervised learning is producing better results than using training data coming from other people as a supervision signal.
- The performance of Person 2 in the supervised case ($\sim 83\%$) is much better than the unsupervised case ($\sim 78\%$). The learned cluster means seem to be much closer to each other and the gaussians overlap more. This can be explained by the lack of data in Class 1 of this person. A better initialization that leverages the training set of other people can improve the result.

Outline

Part I: Gaussian Mixture Models (GMMs) and Expectation Maximization (EM)
1. GMM as a Generative Model
2. Implementation of EM for GMMs
3. Testing EM for GMMs on Synthetic Data
4. Comparing GMMs and K-means

Part II: Maximum A Posteriori (MAP) Estimation

Part III: Modeling the Visual Focus Of Attention (VFOA)
1. Problem Description
2. Supervised Learning Approach
3. Unsupervised Learning using EM
4. MAP Adaptation

MAP Adaptation

Exercice 5

Train 2 GMMs on person 1 and person 2 using the mean $\mu =$ mus_train for initialization. You can use Kmeans initialization for the rest of the parameters.

Solution

```
### BEGIN SOLUTION
pis_init_mean1, mus_init_mean1, sigmas_init_mean1 =learn_parameters_unsupervised(X1, K,
                    max_iter =200, tol =0.001, kmeans_init =True, mus =mus_train)
pis_init_mean2, mus_init_mean2, sigmas_init_mean2 =learn_parameters_unsupervised(X2, K,
                    max_iter =200, tol =0.001, kmeans_init =True, mus =mus_train)
### END SOLUTION

# Evaluate Performance
preds_init_mean1 =predict_label(X1, pis_init_mean1, mus_init_mean1, sigmas_init_mean1)
preds_init_mean2 =predict_label(X2, pis_init_mean2, mus_init_mean2, sigmas_init_mean2)
```

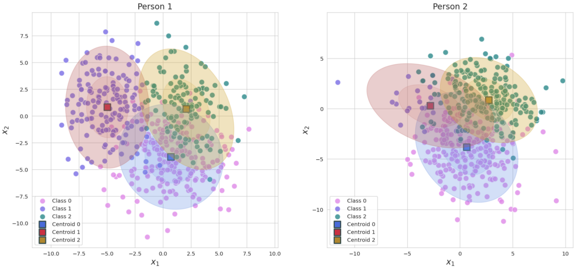
We obtain:

- Accuracy of Person 1: 85.37%
- Accuracy of Person 2: 85.16%

MAP Adaptation

Question 5

Compare the performance of this new initialization with k-means initialization. What do you observe? Why are the results different? Which person was most impacted?



Solution

- We get a big boost in performance for person 2 (85.16% vs 77.64%) and a slight degradation for person 1 (85.37% vs 85.98%). Person 2 is more affected because Class 2 and Class 3 are better captured (from the initialization) and less affected by the lack of data in Class 1.
- Generally speaking, informed initialization improves performance.

MAP Adaptation

Unfortunately, the observations of a test person can be noisy, and different persons may use, on average, different head poses to look at the same targets (e.g. depending on whether they wear glasses). In addition, during an interaction, people may look more or less at different targets, which means we can not necessarily expect a good balance of samples per class.

To address these issues, we can use a prior on model parameters and let the model adapt their values based on the evidence (i.e. test person data) available. This is especially beneficial when we have very little or no data for a particular class, in which case, the prior we set on the gaussian for that class can still provide sensible results.

The MAP adaptation does not affect the Estep. In the Mstep, EM maximizes the function $Q(\theta, \theta^{old}) + \log p(\theta|\Theta)$, where Θ are the parameters defining the prior distributions.

For covenience, we use conjugate distributions for the priors:

- For the categorical distribution over z, the conjugate is a dirichlet ditribution: $\mathrm{Dir}(\pi|\alpha)$.
- For each of the mixtures, we assume a known covariance, and use a gaussian as the conjugate prior over the mean: $p(\mu_k|\beta_k) = \mathcal{N}(\mu_k|\mu_{k0}, \Sigma_{k0})$

MAP Adaptation

The GMM parameters are $\pi = \{\pi_k\}_k$, $\mu = \{\mu_k\}_k$ and $\Sigma = \{\Sigma_k\}_k$.

Again, we assume that the value of Σ is known, and π and μ are random variables with prior distributions:

- Dirichlet Prior on π parameterized by α - i.e. $\text{Dir}(\pi|\alpha)$
- Gaussian Prior on μ parametrized by $\beta = (\mu_0, \Sigma_0)$ - i.e. $\mathcal{N}(\mu|\beta)$.

We have to specify α and β and the initial value of Σ, but the rest will be learnt in the M-step. This should produce the values π^{MAP}, μ^{MAP} and Σ^{new}.

As a refresher, we also write down the following formulas (cf. part 2):

$$\mu_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma_{nk}\mathrm{x}_n \text{ and } \Sigma_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma_{nk}(\mathrm{x}_n - \mu_k^{\text{new}})(\mathrm{x}_n - \mu_k^{\text{new}})^\top,\ N_k = \sum_{n=1}^{N}\gamma_{nk}$$

We also have

$$\pi_k^{MAP} = \frac{\alpha_k + N_k - 1}{\sum_{j=1}^{K}(\alpha_j + N_j - 1)}$$

where α are the parameters of the prior. And μ^{MAP} is given by:

$$\mu_k^{MAP} = \mu_{kp} = \Sigma_{kp}\left(\Sigma_{k0}^{-1}\mu_{k0} + N_k{\Sigma_k^{\text{new}}}^{-1}\mu_k^{\text{new}}\right),\ \text{where } \Sigma_{kp}^{-1} = \Sigma_{k0}^{-1} + N_k{\Sigma_k^{\text{new}}}^{-1}$$

Recall that $\beta_k = (\mu_{k0}, \Sigma_{k0})$ are the parameters of the prior.

MAP Adaptation

Exercice 6

Complete the `GMM_MStep` function below, which performs the M-step of MAP.

Solution

```
def GMM_MStep(X, gammas, alpha, beta):
    ...
    ### BEGIN SOLUTION
    pis_map =gammas.sum(axis =0) +alpha -1
    pis_map /=pis_map.sum()
    ### END SOLUTION
    # Sigma New & mu New
    ...
    # Mu MAP
    for k in range(K):
        priormean =beta['mus'][k]
        priorcov =beta['sigmas'][k]
        N_k =np.sum(gammas[:, k])
        P_prior =np.linalg.inv(priorcov)
        P_data =np.linalg.inv(sigmas_new[k])
        ### BEGIN SOLUTION
        cov_p_k =np.linalg.inv(P_prior +N_k *P_data)
        mus_map[k] =np.matmul(cov_p_k, (np.matmul(P_prior, priormean) +N_k *np.matmul(
                                                                P_data, mus_new[k])))
        ### END SOLUTION

    return pis_map, mus_map, sigmas_new
```

MAP Adaptation

Exercice 7

Complete the `GMM_MAP` function below, which performs MAP estimation.

Solution

```
def GMM_MAP(X, K, alpha, beta, mus =None, sigmas =None, pis =None, kmeans_init =False,
                                                     max_iter =50, tol =0.1, verbose =False):
    # Initialization of the parameters (pi, mu, sigma)
    ...
    # Training
    ...
    # Loop
    for iteration in range(1, max_iter +1):
        for step in ('e', 'm'):
            ### BEGIN SOLUTION
            if step =='e':
                # E-Step
                gammas =GMM_EStep(X, pis, mus, sigmas)
            else:
                # M-Step
                pis, mus, sigmas =GMM_MStep(X, gammas, alpha, beta)
            ### END SOLUTION
            # Compute Lower Bound and Evaluate Log-likelihood
            ...
        # Break if log-likelihood change is lower than tol
        ...
    return pis, mus, sigmas, history
```

MAP Adaptation

Question 7

Compare the classification results obtained through MAP against those obtained in the supervised, or in the fully unsupervised settings. What would influence your choice when setting the parameters α (i.e. Dirichlet prior) and β (i.e. Gaussian prior) ?

Solution

```
pis_map1, mus_map1, sigmas_map1, _ =GMM_MAP(X1, K, alpha, beta, pis =pis_train, mus =
                                              mus_train, sigmas =sigmas_train)
pis_map2, mus_map2, sigmas_map2, _ =GMM_MAP(X2, K, alpha, beta, pis =pis_train, mus =
                                              mus_train, sigmas =sigmas_train)
```

	Supervised	Cheating	Unsupervised	Mean init	MAP
P1	81.91%	86.18%	85.98%	85.37%	84.35%
P2	83.54%	87.80%	77.64%	85.16%	83.94%

- We are slightly worse compared to training-based initialization, but better otherwise. MAP won't necessarily improve performance when the data is good, but it will definitively improve and avoid spurious and non meaningful results when some data are missing. In general, this is a good strategy to exploit when dealing with such cases.
- This is a small dataset, we can achieve better results with more data from more people.
- The variance of the gaussians reflects, for a given person, the expected spread of head poses when looking at a target, and should be set accordingly (e.g. from training data).
- The dirichlet prior should reflect how fast we want to adapt to new data in terms of classes and hence impact the estimated means.
- The prior means is usually used as initialization (cf previous questions).

Thank you for your attention!

Dr. Jean Marc Odobez (https://www.idiap.ch/~odobez)
Samy Tafasca (https://www.idiap.ch/~stafasca)
Idiap Research Institute, Martigny, Switzerland