



MITx 6.86x

Machine Learning with Python-From Linear Models to Deep Learning[Course](#)[Progress](#)[Dates](#)[Discussion](#)[Resources](#)[Course](#) / [Unit 4. Unsupervised Learning...](#) / [Project 4: Collaborative Filtering vi](#)[< Previous](#)

3. Expectation-maximization algorithm


[Bookmark this page](#)

Project due Apr 26, 2023 08:59 -03 Completed


Data Generation Models




Video

 [Download video file](#)

Transcripts

 [Download SubRip \(.srt\) file](#)


 [Download Text \(.txt\) file](#)

Recap of the EM algorithm






Video

 [Download video file](#)

Transcripts


 [Download SubRip \(.srt\) file](#)

 [Download Text \(.txt\) file](#)


Gaussian Mixtures Models for Matrix Completion Continued




Video

 [Download video file](#)



Transcripts

 [Download SubRip \(.srt\) file](#)

 [Download Text \(.txt\) file](#)

▶ 0:00 / 0:00

▶ 1.0x

Video [Download video file](#)**Transcripts** [Download SubRip \(.srt\) file](#) [Download Text \(.txt\) file](#)

Recall the Gaussian mixture model presented in class:

$$P(x|\theta) = \sum_{j=1}^K \pi_j N(x; \mu^{(j)}, \sigma_j^2 I),$$

where θ denotes all the parameters in the mixture (means $\mu^{(j)}$, mixing proportions π_j , and variances σ_j^2). The goal of the EM algorithm is to estimate these unknown parameters by maximizing the log-likelihood of the observed data $x^{(1)}, \dots, x^{(n)}$. Starting with some initial guess of the unknown parameters, the algorithm iterates between E- and M-steps. The E-Step softly assigns each data point $x^{(i)}$ to mixture components. The M-Step takes these soft-assignments as given and finds a new setting of the parameters by maximizing the log-likelihood of the weighted dataset (expected complete log-likelihood).

Implement the EM algorithm for the Gaussian mixture model described above. To this end, you will write the functions `estep`, `mstep` and `run` in `naive_em.py`. In our notation,

- `X`: an (n, d) Numpy array of n data points, each with d features
- `K`: number of mixture components
- `mu`: (K, d) Numpy array where the j^{th} row is the mean vector $\mu^{(j)}$
- `p`: $(K,)$ Numpy array of mixing proportions $\pi_j, j = 1, \dots, K$
- `var`: $(K,)$ Numpy array of variances $\sigma_j^2, j = 1, \dots, K$

The convergence criteria that you should use is that the improvement in the log-likelihood is less than 10^{-6} multiplied by the absolute value of the new log-likelihood. In slightly more algebraic notation, the convergence criteria is $|\text{new log-likelihood} - \text{old log-likelihood}| \leq 10^{-6} \cdot |\text{new log-likelihood}|$.

Your code will output updated versions of a `GaussianMixture` (with means `mu`, variances `var`, and mixing proportions `p`) as defined in `common.py` as well as an (n, K) Numpy array `post`, where `post[i, j]` is the posterior probability $p(j|x^{(i)})$, and `LL` which is the log-likelihood of the weighted dataset.

Implementing E-step

1.0/1.0 point (graded)

Write a function `estep` that performs the E-step of the EM algorithm

Available Functions: You have access to the NumPy python library as `np`, to the `GaussianMixture` class and to typing annotation `typing.Tuple` as `Tuple`

```
1 def estep(X: np.ndarray, mixture: GaussianMixture) -> Tuple[np.ndarray, float]:
2     """E-step: Softly assigns each datapoint to a gaussian component
3
4     Args:
5         X: (n, d) array holding the data
6         mixture: the current gaussian mixture
7
8     Returns:
9         np.ndarray: (n, K) array holding the soft counts
10        for all components for all examples
11        float: log-likelihood of the assignment
12    """
13    from scipy.stats import multivariate_normal
14    n, d = X.shape
15    k = mixture.mu.shape[0]
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

CORRECT

Submit

You have used 4 of 50 attempts

Implementing M-step

1.0/1.0 point (graded)

Write a function `mstep` that performs the M-step of the EM algorithm

Available Functions: You have access to the NumPy python library as `np`, to the `GaussianMixture` class and to typing annotation `typing.Tuple` as `Tuple`

Test results

CORRECT

Submit

You have used 5 of 50 attempts

Implementing run

1.0/1.0 point (graded)

Write a function `run` that runs the EM algorithm. The convergence criterion you should use is the one defined above.

Available Functions: You have access to the NumPy python library as `np`, to the `GaussianMixture` class and to typing annotation `typing.Tuple` as `Tuple`. You also have access to the `estimate` function you have just implemented

```
1 def run(X: np.ndarray, mixture: GaussianMixture,
2         post: np.ndarray) -> Tuple[GaussianMixture, np.ndarray, float]:
3     """Runs the mixture model
4
5     Args:
6         X: (n, d) array holding the data
7         post: (n, K) array holding the soft counts
8              for all components for all examples
9
10    Returns:
11        GaussianMixture: the new gaussian mixture
12        np.ndarray: (n, K) array holding the soft counts
13                   for all components for all examples
14        float: log-likelihood of the current assignment
15    """
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

CORRECT

[Getting v](#)[< Previous](#)[Next >](#)[? Stuck on log-likelihood in E-step](#)

edX

[About](#)[Affiliates](#)[edX for Business](#)[Open edX](#)[Careers](#)[News](#)

Legal

[Terms of Service & Honor Code](#)[Privacy Policy](#)[Accessibility Policy](#)[Trademark Policy](#)[Sitemap](#)[Cookie Policy](#)[Do Not Sell My Personal Information](#)

Connect

[Blog](#)[Contact Us](#)[Help Center](#)[Security](#)[Media Kit](#)