3. Expectation–maximization algorithm

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

Data Generation Models

0:00 / 0:00 1.0x

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Recap of the EM algorithm

0:00 / 0:00

1.0x

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Gaussian Mixtures Models for Matrix Completion Continued

0:00 / 0:00

1.0x

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, a
goal of the EM algorithm is to estimate these unknown parameters by maximizing the lo
observed data $x^{(1)}, \ldots, x^{(n)}$. Starting with some initial guess of the unknown parameter
between E- and M-steps. The E-Step softly assigns each data point $x^{(i)}$ to mixture cor
takes these soft-assignments as given and finds a new setting of the parameters by ma
likelihood of the weighted dataset (expected complete log-likelihood).

Implement the EM algorithm for the Gaussian mixture model desribed above. To this en
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 π_j, $j = 1, \ldots, K$
- `var`: $(K,)$ Numpy array of variances σ_j^2, $j = 1, \ldots, K$

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

Your code will output updated versions of a `GaussianMixture` (with means `mu`, varian
proportions p) as defined in `common.py` as well as an (n, K) Numpy array `post`, where p
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 `Gau`
and to typing annotation `typing.Tuple` as `Tuple`

```
def estep(X: np.ndarray, mixture: GaussianMixture) -> Tuple[np.ndarray, fl
    """E-step: Softly assigns each datapoint to a gaussian component

    Args:
        X: (n, d) array holding the data
        mixture: the current gaussian mixture

    Returns:
        np.ndarray: (n, K) array holding the soft counts
            for all components for all examples
        float: log-likelihood of the assignment
    """
    from scipy.stats import multivariate_normal
    n, d = X.shape
    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 `Gau`
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 shoul
above.

Available Functions: You have access to the NumPy python library as `np`, to the `Gau`
and to typing annotation `typing.Tuple` as `Tuple`. You also have access to the `este`
functions you have just implemented

```
def run(X: np.ndarray, mixture: GaussianMixture,
        post: np.ndarray) -> Tuple[GaussianMixture, np.ndarray, float]:
    """Runs the mixture model

    Args:
        X: (n, d) array holding the data
        post: (n, K) array holding the soft counts
            for all components for all examples

    Returns:
        GaussianMixture: the new gaussian mixture
        np.ndarray: (n, K) array holding the soft counts
            for all components for all examples
        float: log-likelihood of the current assignment
    """
```

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

Correct

Test results

CORRECT

Getting v < Previous ng Next >

? Stuck on log-likelihood in E-step