5. Bayesian Information Criterion

Extension Note: Project 4 due date has been extended by 1 more day to **August 22 23:59UTC** .

So far we have simply set the number of mixture components K but this is also a parameter that we must estimate from data. How does the log-likelihood of the data vary as a function of K assuming we avoid locally optimal solutions?

To compensate, we need a selection criterion that penalizes the number of parameters used in the model. The Bayesian information criterion (BIC) is a criterion for model selection. It captures the tradeoff between the log-likelihood of the data, and the number of parameters that the model uses. The BIC of a model M is defined as:

$$\mathrm{BIC}\,(M) = l - \frac{1}{2} p \log n$$

where l is the log-likelihood of the data under the current model (highest log-likelihood we can achieve by adjusting the parameters in the model), p is the number of adjustable parameters, and n is the number of data points. This score rewards a larger log-likelihood, but penalizes the number of parameters used to train the model. In a situation where we wish to select models, we want a model with the the highest BIC.

Implementing the Bayesian Information Criterion

1.0/1.0 point (graded)

Fill in the missing Bayesian Information Criterion (BIC) calculation (`bic` function) in `common.py` .

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

```
        X: (n, d) array holding the data
        mixture: a mixture of spherical gaussian
        log_likelihood: the log-likelihood of the data

    Returns:
        float: the BIC for this mixture
    """
    n, d = X.shape
    mu, var, p = mixture
    n_para = mu.shape[0] * mu.shape[1] + var.shape[0] + p.shape[0] - 1
    BIC = log_likelihood - 1/2 * n_para * np.log(n)
    return BIC
```

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

Correct

```
def bic(X: np.ndarray, mixture: GaussianMixture,
        log_likelihood: float) -> float:
    """Computes the Bayesian Information Criterion for a
    mixture of gaussians

    Args:
        X: (n, d) array holding the data
        mixture: a mixture of spherical gaussian
        log_likelihood: the log-likelihood of the data

    Returns:
        float: the BIC for this mixture
    """
    n, _ = X.shape
    K, d = mixture.mu.shape

    return log_likelihood - (K * d + K + K - 1) / 2 * np.log(n)
```

Test results

CORRECT

See full output

See full output

Solution:

The Bayesian Information Criterion for a mixture of spherical Gaussians is:

$$BIC\left(D;\theta\right) = l\left(D;\theta\right) - \frac{k\left(d+2\right)-1}{2}\log\left(n\right)$$

Submit

You have used 4 of 20 attempts

Answers are displayed within the problem

Picking the best K

1.0/1.0 point (graded)

Find the best K from $[1,2,3,4]$ on the toy dataset. This will be the K that produces the optimal BIC score. Report the best K and the corresponding BIC score. Measure the BIC on EM models, only. Does the criterion select the correct number of clusters for the toy data?

$\text{Best K} =$ 3 ✔ **Answer:** 3

$\text{Best BIC} =$ -1169.2589347355092 ✔ **Answer:** -1169.2589

Grader note: While the best BIC should be a negative value, due to earlier grader error, we have corrected the grader to accept both the positive and the negative value.

Solution:

Code:

```
def run_with_bic():
    max_bic = None
    for K in range(1, 5):
        max_ll = None
        best_seed = None
        for seed in range(0, 5):
            mixture, post = common.init(X, K, seed)
            mixture, post, ll = naive_em.run(X, mixture, post)
            if max_ll is None or ll > max_ll:
                max_ll = ll
                best_seed = seed

        mixture, post = common.init(X, K, best_seed)
        mixture, post, ll = naive_em.run(X, mixture, post)
        bic = common.bic(X, mixture, ll)
        if max_bic is None or bic > max_bic:
            max_bic = bic
        title = "EM for K=, seed=, ll=, bic=".format(K, best_seed, ll, bic)
        print(title)
        common.plot(X, mixture, post, title)
```

K	BIC
1	-1315.5056
2	-1195.0397
3	-1169.2589
4	-1180.0121

From the BIC values above, the best K is 3, which seems to fit the toy data.

Submit

You have used 1 of 10 attempts

Answers are displayed within the problem

Discussion

Show Discussion

Topic: Unit 4 Unsupervised Learning (2 weeks) :Project 4: Collaborative Filtering via Gaussian Mixtures / 5. Bayesian Information Criterion