UDACITY

PROJECT

Build a Sign Language Recognizer

A part of the Artificial Intelligence Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Congratulations!

You're done, you may want to update the value of `p` if you'd like.

To be honest, I think this confusion may be our fault, but still leads students to research on their own, and that's a key factor to have.

You have done a great project and shown great interest, therefore you earned this:

PART 1: Data

1. Student provides correct alternate feature sets: delta, polar, normalized, and custom.
2. Student passes unit tests.
3. Student provides a reasonable explanation for what custom set was chosen and why (Q1).

PART 2: Model Selection

1. Student correctly implements CV, BIC, and DIC model selection techniques in “my_model_selectors.py”.
2. Student code runs error-free in notebook, passes unit tests and code review of the algorithms.
3. Student provides a brief but thoughtful comparison of the selectors (Q2).

To be fair, this has been a major point of confusion, you did your homework so I'm passing this 😉

I think there is a confusion with the `number of data points` with the `number of features`, wich the last one is the one that we care for `p`.

`N` is the number of data points, `f` is the number of features:

```
N, f = self.X.shape
```

Having `m` as the `num_components`, The free parameters `p` are a sum of:

- The free transition probability parameters, which is the size of the transmat matrix less one row because they add up to 1 and therefore the final row is deterministic, so `m*(m-1)`
- The free starting probabilities, which is the size of startprob minus 1 because it adds to 1.0 and last one can be calculated so `m-1`
- The number of means, which is `m*f`
- Number of covariances which is the size of the covars matrix, which for "diag" is `m*f`

All of the above is equal to:

```
p = m^2 +2mf-1
```

Finally, the BIC equation is:

```
BIC = -2 * logL + p * logN
```

PART 3: Recognizer

1. **Student implements a recognizer in "my_recognizer.py" which runs error-free in the notebook and passes all unit tests**
2. **Student provides three examples of feature/selector combinations in the submission cells of the notebook.**
3. **Student code provides the correct words within <60% WER for at least one of the three examples student provided.**
4. **Student provides a summary of results and speculates on how to improve the WER.**

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ