UDACITY

PROJECT

# Build a Sign Language Recognizer

A part of the Artificial Intelligence Nanodegree Program

PROJECT REVIEW

CODE REVIEW 8

NOTES

SHARE YOUR ACCOMPLISHMENT!

## Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Congratulations! You made a *remarkable* project.

Just considering updating the value of `p` in BIC, and you'll have a perfect *way* beyond expectations project.

We're waiting for the next awesome submission 😃

### 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).**

👍 Great rescaled features!

### 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).**

👏 Awesome job with your job selectors.

There is a tiny discrepancy with `p` in BIC.

For BIC, there is more one way to calculate `p`, below is a simple one:

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

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

If `m` is num_components:

```
p = m^2 +2mf−1
```

👍 Nice analysis on model selectors!

You may also want to know that SelectorCV is the only selector that tests unseen data. BIC tries to avoid *overfitting* by penalizing large number of features (and states).
DIC is useful because evaluates the performance of competing words, it chooses models that get low score on competing words. This is useful because penalizes the model when it confuses with other words.

👏 Awesome plot for time and states analysis!

## 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.**

👏 Very good section!

You achieved awesome 48% of WER, this is truly awesome ⭐

I would ask to rerun this after updating the `p` parameter, but since it's so good you won't need to 😉

👏 Very good plots!

NIce answer on improving the WER using n-gram. The takeaway for improving WER is that words are influenced by their **context** (words before and after). For example is highly likely to see the word *Union* after the word *Soviet*.

RESUBMIT

DOWNLOAD PROJECT

8 CODE REVIEW COMMENTS ›