PROJECT

Build a Sign Language Recognizer

A part of the Artificial Intelligence Program

PROJECT REVIEW

CODE REVIEW 4

NOTES

my_model_selectors.py 4

```
import math
import statistics
import warnings

import numpy as np
from hmmlearn.hmm import GaussianHMM
from sklearn.model_selection import KFold
from asl_utils import combine_sequences


class ModelSelector(object):
    '''
    base class for model selection (strategy design pattern)
    '''

    def __init__(self, all_word_sequences: dict, all_word_Xlengths: dict, this_word: str,
                 n_constant=3,
                 min_n_components=2, max_n_components=10,
                 random_state=14, verbose=False):
        self.words = all_word_sequences
        self.hwords = all_word_Xlengths
        self.sequences = all_word_sequences[this_word]
        self.X, self.lengths = all_word_Xlengths[this_word]
        self.this_word = this_word
        self.n_constant = n_constant
        self.min_n_components = min_n_components
        self.max_n_components = max_n_components
        self.random_state = random_state
        self.verbose = verbose

    def select(self):
        raise NotImplementedError

    def base_model(self, num_states):
        # with warnings.catch_warnings():
        warnings.filterwarnings("ignore", category=DeprecationWarning)
        # warnings.filterwarnings("ignore", category=RuntimeWarning)
        try:
            hmm_model = GaussianHMM(n_components=num_states, covariance_type="diag", n_iter=1000,
                                    random_state=self.random_state, verbose=False).fit(self.X, self.lengths)
            if self.verbose:
                print("model created for {} with {} states".format(self.this_word, num_states))
            return hmm_model
        except:
            if self.verbose:
                print("failure on {} with {} states".format(self.this_word, num_states))
            return None


class SelectorConstant(ModelSelector):
    """ select the model with value self.n_constant

    """

    def select(self):
        """ select based on n_constant value

        :return: GaussianHMM object
        """
        best_num_components = self.n_constant
        return self.base_model(best_num_components)


class SelectorBIC(ModelSelector):
    """ select the model with the lowest Bayesian Information Criterion(BIC) score

    http://www2.imm.dtu.dk/courses/02433/doc/ch6_slides.pdf
    Bayesian information criteria: BIC = -2 * logL + p * logN
    """

    def select(self):
        """ select the best model for self.this_word based on
        BIC score for n between self.min_n_components and self.max_n_components

        :return: GaussianHMM object
        """
        warnings.filterwarnings("ignore", category=DeprecationWarning)

        # TODO implement model selection based on BIC scores
        l = math.inf
        model = None
        for nc in range(self.min_n_components, self.max_n_components + 1):
            try:
                hmm_model = GaussianHMM(n_components=nc, covariance_type="diag", n_iter=1000,
                                        random_state=self.random_state, verbose=self.verbose)
                hmm_model.fit(self.X, self.lengths)
                d = hmm_model.n_features
                n = hmm_model.n_components
                N = len(self.lengths)
                p = n**2 + 2*n*d - 1
```

AWESOME

The no. of free parameters has been calculated correctly. This reflects your understanding of the topic and the research done 👍

Formula is perfectly implemented!

```
                logL = hmm_model.score(self.X, self.lengths)
                bic = -2*logL + p*np.log(N)
                if bic < l:
                    l = bic
                    model = hmm_model
            except:
                pass
        return model


class SelectorDIC(ModelSelector):
    ''' select best model based on Discriminative Information Criterion

    Biem, Alain. "A model selection criterion for classification: Application to hmm topology optimization."
    Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on. IEEE, 2003.
    http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.6208&rep=rep1&type=pdf
    DIC = log(P(X(i)) - 1/(M-1)SUM(log(P(X(all but i))
    '''

    def select(self):
        warnings.filterwarnings("ignore", category=DeprecationWarning)

        # TODO implement model selection based on DIC scores
        l = -math.inf
        model = None
        for nc in range(self.min_n_components, self.max_n_components + 1):
            try:
                hmm_model = GaussianHMM(n_components=nc, covariance_type="diag", n_iter=1000,
                                        random_state=self.random_state, verbose=self.verbose)
                hmm_model.fit(self.X, self.lengths)
                logL = hmm_model.score(self.X, self.lengths)
                sc = 0
                for word, (x, lengths) in self.hwords.items():
                    if word != self.this_word:
                        sc += hmm_model.score(x, lengths)
                mean_other_words = sc / (len(self.hwords) - 1)
                dic = logL - mean_other_words
```

AWESOME

Very impressive! The score "P(X(all but i))" was a bit difficult to calculate and you nailed it 👍

Formula for DIC has been perfectly implemented!

```
                if dic > l:
                    l = dic
                    model = hmm_model
            except:
                pass
        return model
```

```


class SelectorCV(ModelSelector):
    ''' select best model based on average log Likelihood of cross-validation folds

    '''

    # @property
    def select(self):
        warnings.filterwarnings("ignore", category=DeprecationWarning)
        # TODO implement model selection using CV
        model = None
        low_score = -math.inf
        for n_comp in range(self.min_n_components, self.max_n_components + 1):
            # get the model & its average score
            score, hmm_model = self.avg_score(n_comp)
            # find the model with highest average score
```

SUGGESTION

Inline comments are used to make the code more readable

```
            if score > low_score:
                low_score, model = score, hmm_model
        return model

    def avg_score(self, n_comp):
        # Collect the average score of the model for its different folds.
        scores = []
        hmm_model = None
        n_split = 2 if len(self.sequences) < 3 else 3
        split_mtd = KFold(n_splits=n_split)
        for cv_train_idx, cv_test_idx in split_mtd.split(self.sequences):
            x_train, len_train = combine_sequences(cv_train_idx, self.sequences)
            x_test, len_test = combine_sequences(cv_test_idx, self.sequences)
```

AWESOME

Good use of combine_sequences utility 👍

```
            try:
                # not using the base_model() as it uses verbose = False for all cases.
                hmm_model = GaussianHMM(n_components=n_comp, covariance_type="diag", n_iter=1000,
                                        random_state=self.random_state, verbose=self.verbose)
                hmm_model.fit(x_train, len_train)
                scores.append(hmm_model.score(x_test, len_test))
            except:
                pass
        avg = np.mean(scores) if len(scores) else -math.inf
        return avg, hmm_model

```

▸ my_recognizer.py

▸ asl_utils.py

▸ asl_recognizer.html

RETURN TO PATH

Student FAQ