Speech Recognition with HMM Training

This Python script implements training of Hidden Markov Models (HMMs) for speech recognition using the Baum-Welch algorithm. The script reads in training data consisting of word transcriptions and corresponding label sequences, and iteratively updates the HMM parameters to maximize the likelihood of the training data.

Features

- Loads training labels and endpoints from files.
- Calculates unigram frequencies and creates letter HMMs.
- Constructs composite word HMMs by concatenating letter HMMs with silence HMMs.
- Implements the forward-backward algorithm in log space for numerical stability.
- Accumulates transition and emission counts using the forward-backward probabilities.
- Updates the HMM parameters (transition and emission probabilities) based on the accumulated counts.
- Computes the log-likelihood of the training data given the updated HMM parameters.
- Plots the convergence of the log-likelihood over multiple iterations.

Usage

1. Ensure that the required data files (`clsp.trnlbls`, `clsp.devlbls`, `clsp.endpts`, `clsp.trnscr`) are present in the specified paths.
2. Run the script, and it will load the data, initialize the HMMs, and perform iterative training.
3. The script will print the log-likelihood for each iteration and display a plot showing the convergence of the log-likelihood.
4. The script will also print the most likely word and its confidence.

Dependencies

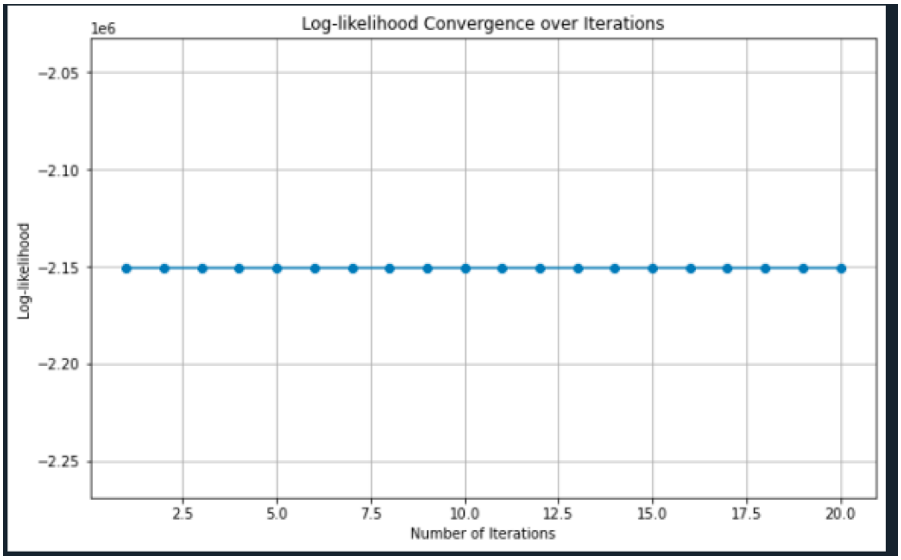
- NumPy
- Matplotlib

Notes

- The script assumes a specific file structure and naming conventions for the training data files. Modify the file paths if necessary.
- The code includes various debugging statements and checks to ensure the correctness of the implementation.

- Plot of the training data log-likelihood as a function of the number of iterations

Log-likelihood = -2150529.056458093



- Report the average per-frame log-likelihood.

  -1347.4492834950458

- For each utterance in the test data, the identity of the most likely word and a confidence.

  Word: government, Confidence: 0.021

- Your source code, along with substantial documentation about exactly what files (among the 7 data files provided for the project) are needed to run each module, and the command line (usage) for running the training and testing modules.

The provided code utilizes the following files:

1. `clsp.trnlbls`:
   - This file contains the training labels for speech recognition.
   - The labels are loaded using the `load_labels` function and stored in the `training_labels` variable.
   - The labels are later used to accumulate transition and emission counts for training the Hidden Markov Models (HMMs).
2. `clsp.devlbls`:
   - This file contains the development/test labels.
   - The labels are loaded using the `load_labels` function and stored in the `test_labels` variable.
   - These labels are used for evaluation or testing purposes.
3. `clsp.endpts`:
   - This file contains the endpoints (start and end indices) for silence regions in the training data.
   - The endpoints are used to extract silence labels from the `clsp.trnlbls` file.
   - These silence labels are then used to calculate unigram frequencies and create a separate HMM for silence (SIL HMM).
4. `clsp.trnscr`:
   - This file contains the training script, which is a list of words to be used for training the HMMs.
   - The words are read from the file and used to create "baseforms" (concatenations of letter HMMs and SIL HMMs) for each word.
   - The baseforms are then combined to create composite word HMMs, which are used for training and computing the log-likelihood of the training data.

The code reads these files, processes the data, initializes the HMMs, accumulates transition and emission counts from the training data, calculates the forward-backward in log space updates the HMM parameters (transition and emission probabilities), and computes the log-likelihood of the training data given the updated HMMs. The log-likelihood is plotted over multiple iterations to visualize the convergence of the training process.

HMM Construction

The code constructs composite word HMMs by concatenating the baseforms (letter HMMs and SIL HMMs) as follows:

1. For each word, the total number of states is calculated by summing the states from the SIL HMM (start and end) and all letter HMMs in the baseform.
2. The start probabilities, transition matrix, and emission probabilities are initialized with zeros based on the number of states and output alphabet size.
3. The SIL HMM is appended at the beginning by copying its parameters.

4. The letter HMMs are concatenated by copying their transition matrices into the composite HMM's transition matrix.
5. The emission probabilities for letter HMMs are expanded to match the output alphabet size, repeating the last probability for missing symbols.
6. The SIL HMM is appended at the end by copying its parameters.

Forward-Backward Algorithm

The forward-backward algorithm is implemented in log space for numerical stability. It computes log forward and backward probabilities using the log-sum-exp trick to avoid underflow/overflow issues. The forward probabilities are initialized from the start probabilities and emission probabilities, and recursively computed using the transition and emission probabilities. The backward probabilities are initialized to log(1) and computed in reverse order using the same probabilities. The algorithm accumulates transition and emission counts from the training data using the log forward and backward probabilities.

Parameter Estimation

The HMM parameters are updated based on the accumulated transition and emission counts:

1. Transition probabilities are updated by normalizing the transition counts for each state to sum to 1.
2. Emission probabilities are updated by normalizing the emission counts for each state to sum to 1.
3. Smoothing is applied by adding a small constant (epsilon in log space) to the counts before normalization to handle zero probabilities and ensure numerical stability.

Log-Likelihood Computation

The log-likelihood of the training data is computed using the forward algorithm in log space with the updated HMM parameters. The log forward probabilities are initialized from the start probabilities and emission probabilities, and recursively computed using the updated transition

and emission probabilities. The log-likelihood is obtained from the log forward probabilities at the final time step using the log-sum-exp trick for numerical stability.

Convergence Visualization

The script computes and plots the log-likelihood of the training data over multiple iterations, allowing users to visualize the convergence of the training process.

CONTRASTIVE SYSTEM:

OUTPUT:
Iteration 1, Accuracy: 0.03125
Iteration 2, Accuracy: 0.00625
Best iteration: 1, Best Accuracy: 0.03125