67355 Introduction to Speech Processing

Exercise 4

In the following exercise you will implement the CTC algorithm.
The exercise should be done **in pairs** and is to be submitted via moodle by the deadline appearing under the submission box.
See submission guidelines for further instructions

1 Connectionist Temporal Classification

In this exercise you will implement the CTC loss in Python. CTC calculates the probability of a specific labeling given the model's output distribution over phonemes. Formally, CTC calculates $P(\boldsymbol{p}|\mathbf{x})$ where $\mathbf{x} = [x_1, x_2, \dots, x_T]$ is an input sequence of acoustic features, $\boldsymbol{p} = [p_1, p_2, \dots, p_{|\boldsymbol{p}|}]$ is a sequence of transcription phonemes, and y is a sequence of network outputs, that is, y_k^t can be interpreted as the probability of observing label k at time t.

Recall, to calculate the aforementioned probability, we first set,

$$\boldsymbol{z} = [\epsilon, p_1, \epsilon, p_2, \epsilon, \dots, p_{|\boldsymbol{p}|}, \epsilon]. \tag{1}$$

Then, we define $\alpha_{s,t}$ to be the probability of the subsequence $z_{1:s}$ after t time steps. We can calculate α using the following initialization:

$$\begin{aligned} \alpha_{1,1} &= y_\epsilon^1 \\ \alpha_{2,1} &= y_{z_1}^1 \\ \alpha_{s,1} &= 0, \qquad \forall s > 2, \end{aligned} \tag{2}$$

and the following dynamic programming:

$$\alpha_{s,t} = \begin{cases} (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot y_{z_s}^t & z_s = \epsilon or z_s = z_{s-2} \\ (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot y_{z_s}^t & \text{else.} \end{cases} \tag{3}$$

2 Instructions

In this exercise, assume you are given a sequence of phonemes $\boldsymbol{p}$ and the network's output y. In words, y is a matrix with the shape of $T \times K$ where T is the number of time steps, and K is the amount of phonemes. Each row i of $\boldsymbol{y}$ is a distribution over K phonemes at time i.

Your goal is to implement the CTC function to calculate $P(\boldsymbol{p}|\boldsymbol{x})$ using the above equations. Your code should get 3 arguments:

1. A path to a 2D numpy matrix of network outputs ($\boldsymbol{y}$). This should be loaded using `numpy.load`.
2. A string of the labeling you wish to calculate the probability for (e.g., "aaabb" means we want the probability of aaabb).
3. A string specifying the possible output tokens (e.g., for an alphabet of [a,b,c] the string should be "abc").

Overall, your code should run with the following command: "`python ex4.py /some/path/to/mat.npy aaabb abc`". For your convinience, we attach also an example of inputs.
Expected output: A single printed line containing only $P(\boldsymbol{p}|\boldsymbol{x})$ rounded up to 3 decimal points, specifically use the following function to print your prediction:

```
def print_p(p: float):
    print("%.3f" % p)
```

It is crucial that you make sure that you only print a single line! we will be using 'diff' to evaluate your performance hence failing to follow the instructions could decrease your grade drastically.

3 Submission Guidelines

- Submission should be done in pairs only.
- All used code pieces should be submitted and tested on the school's computers using the following python(3.9) constraints:
  - torchaudio==0.13.1
  - torch==1.13.1
  - soundfile==0.12.1
  - librosa==0.10.0.post2
  - numpy==1.23.5
  - scipy==1.9.3
  - scikit-learn==1.2.0
  - pytorch-lightning==2.0.2
- Your submission MUST include README.txt file containing a single line of the following format:

  ```
  <ID1>_<ID2>
  ```

- Please submit a single zip/tar file containing all relevant files.