Lab 9: Keyword recognition

Part 1. Making a digit recognizer

In this section we will design a simple spoken digit recognizer, based on Dynamic Time Warping (DTW). In order to make such a system we need to first collect some data, and then design a DTW routine that can compare new inputs with templates for each digit.

To start with make a set of data that will be used here. Make a 4-5 recordings of yourself speaking each of the ten digits (0 to 9). We will use one recording from each digit as the template, and the rest at testing data. In order to not spend too much time collecting the data, record all these utterances in a single (long) sound file. Use your voice activity detector to split that file into the individual spoken digits.

In order to design a digit recognizer we will take a spoken input of a digit and compare it to each digit's template. By finding which template is the most similar we can classify the input as belonging to that template's digit. In order to measure the distance between the two sequences we have to use DTW on an appropriate feature space.

Decide which feature to use to represent your speech signals. It can be any feature that we used in the past (e.g. some type of an STFT, MFCCs, etc). When comparing a template with a new input you need to perform the following steps:

1. Compute the distance matrix between all the features of each input. This will be a M by N matrix in which the (i, j) element will represent the distance between the i-th frame of the template and the j-th frame of the input. We will use the *cosine distance* which is defined as:

$$D(\mathbf{a},\mathbf{b}) = 1 - \frac{\sum a_i b_i}{\sqrt{\sum a_i^2}\sqrt{\sum b_i^2}}$$

2. Once you obtain the distance matrix, you need to compute the cost matrix that encodes the cost of passing through a node given a previously optimal path. We will use the local constraint that to reach node (i, j) you can either come from nodes $(i{-}1, j{-}1)$, $(i, j{-}1)$ or $(i{-}1, j)$.

3. Starting from the first element of the matrix $(1,1)$, and for each element of the cost matrix you will need to perform the following steps. For node (i, j) you need to examine the nodes from which you can reach it – these will be nodes $(i{-}1, j{-}1)$, $(i, j{-}1)$ or $(i{-}1, j)$ – and see which one has the lowest cost. Therefore, reaching that node from the optimal path will have the cost of the optimal preceding node plus the distance that corresponds to being at node (i, j). Iterate until you calculate the cost of passing through every node. As you do that, for each node keep track of which of the three preceding nodes was the optimal one.

4. Now you can backtrack and find the optimal path. Start from the final point of the cost matrix and find the node from which you arrived there (it will be the same one that had the lowest cost above). Once you get to that node, repeat this process until you reach the beginning indexes of the two sequences. The path that you took in this process will be the optimal path that aligns the two sequences.

5. The distance between the two sequences will be the cost of being at the final node. Use this to perform the digit classification.

Part 2. Making a voice-driven phone dialer

Suppose you just started working for an auto company and the first thing they ask you is to make a hands-free interface for their cars so that people can dial in their friends by voice. During setup, the users speak the name of a contact and then associate it with a number to call. Make a system for which you use the full name of 4-5 of your friends, so that when you speak their name the system recognizes it.