Task 9: Biological Applications

Data

This practical explores the application of Hidden Markov models to biological data. To start, download the bio dataset and look at the file. The format is different to the dice dataset you worked with before. The observed sequence starts with a hash (which you should otherwise ignore) and the corresponding hidden sequence follows in a row below. Each observed-hidden sequence pair is separated by an empty line. You can read in the dataset in Python by calling the method `load_bio_data(path)` from `utils.markov_models`.

The observed sequence consists of a series of amino acids. As you may remember from biology classes, amino acids typically have three letter abbreviations, like Leu for leucine. In our dataset these more common names have been replaced with single letters. You can find the mapping in `utils/markov_models/data_loader.py` class if you are curious.

The sequences of amino acids correspond to **transmembrane proteins**: i.e., protein which are located so that they cross the membrane. Each of the amino acids is associated with a hidden `Feature` state, which indicates whether the amino acid is inside the cell, outside the cell, or in the membrane. From the application point of view, the membrane state is the most interesting one. It is actually rather difficult to distinguish between the inside and outside states, so low overall accuracy is to be expected.

This data was introduced as part of the following paper,
https://pubmed.ncbi.nlm.nih.gov/11152613/ - full paper requires that you login in via Raven/Shibboleth.

Step 1: Adapt HMM for Biological Task

Adapt your code from Tasks 7 and 8 to build a Hidden Markov Model of the amino acid sequences and to predict hidden states using the Viterbi algorithm. What are the hidden and observed states in this task?

As in Task 8, you can use the provided code that splits the data into train-dev-test. What results do you get if you vary the random seed for the splitting? After varying the seed, you should understand why you should implement cross-validation.

Step 2: Semi-Supervised Learning with Self-Training

You might have noticed that we only have a small number of annotations available to train our HMM for the biological task. This makes it difficult

to estimate the inherent distribution of the data correctly and subsequently affects the performance of the HMM on unseen data. While obtaining a larger number of annotations is often difficult and expensive, obtaining additional observed sequences is usually simpler. You will explore a simple semi-supervised approach of incorporating additional data of only observed sequences to refine your HMM model. By iteratively re-training your model using additional observed sequences labelled with hidden states annotated by the model itself, the self-trained model will incorporate information of the additional observed sequences into its probabilities. Self-training of your HMM model consists of following steps:

0. Train an HMM model with the available labelled training data as done in the first part of the task.
1. Use the trained model to predict the hidden sequences of all available unlabelled data.
2. Merge the labelled data with the new pseudo-labelled data.
3. Train a new HMM model on the merged data. Repeat Step 1 (for the now pseudo-labelled data) to 3 for a total of t iterations.

Download the unlabelled data, created artificially by another model, and place it into `data/markov_models`. Compute recall, precision, and F_1 after each iteration (on the dev set) and plot them using the methods provided in Tick 3.

What will happen if you use the observed sequences of the labelled training data as unlabelled data for the self-training algorithm? What about the development data?

Currently, the algorithm is re-labeling all instances at each iteration. What are possibly better ways of selecting instances to label and re-label?

Starred Tick (Optional)

If you think about the biological task, you will realize that our first-order HMM cannot completely capture the structure of the data. What is missing, in terms of state sequence information? What's more, if you analyse the dataset, you will see that the length of the 'within membrane' sequence is relatively constrained. Find the minimum and maximum length. Why does this make sense in biological terms?

How could we adapt the simple HMM approach to take account of this structure? This is quite challenging: probably the easiest approach (though not the most efficient) is to generate the n-best sequence of hidden states using Viterbi (instead of just the most probable sequence) and to filter accoring to the other constraints. If you do have a go at this exercise, please let us know what you did and what results you get. Because of the small size of the dataset, you will have to be careful about methodology!

If you look at the papers written by the creators of the dataset we're using you will see that the way HMMs are used there is considerably more elaborate than the experiments we're carrying out.