Tutorial - Exercise sheet 8

Pattern and Speech Recognition

Introduction

During this tutorial, you will train a decision tree on the duration of the ARCTIC corpus. We already used this corpus in previous tutorials but it was about cepstral coefficients. For this tutorial, the leaves of the decision tree will contain the **mean** duration associated with the cluster.

1 Loading the data [1pt + 2 bonus points]

Ex. 1 — Download the data set from `https://www.coli.uni-saarland.de/~slemaguer/teaching/corpus_duration.csv` and the question file from `https://www.coli.uni-saarland.de/~slemaguer/teaching/questions.hed`

Ex. 2 — Plot the distribution of the durations (**do not take the properties into account just the last column**). What can you conclude about it ?

Ex. 3 — [**BONUS**] Is a GMM adapted to duration modeling ? Justify. How many components will you use? Justify

Ex. 4 — [**BONUS**] What is the main differences between the GMM modeling and the decision trees ? Can we mix both modeling ? Explain how.

Ex. 5 — Divide the dataset into 3 subsets : a *training* set which is containing 90% of the original set, a *validation* and a *test* set which are containing **each** 5% of the original set

2 Decision tree clustering [6pt]

To achieve the decision tree clustering, we are going to use the two corpora and *entropy impurity criterion.*

The *entropy impurity criterion* will be used to determine which leaf is going to split and to select the question. This is what we called the **splitting criterion**.

We also need a *stopping criterion.* In our case, we are going to *over-training detection.* For that, we need the validation corpus and a distance measure. In our case we are going to use the RMSE that you have implemented before. The *stopping criterion* is based on the fact that **the mean of the RMSE** should decrease. If it starts to increase, it means we are facing an over-training and should stop. In our case, we are considering an over-training when the differences between the current and previous means of RMSE should be greater than an epsilon.

2.1 Preliminary work

Ex. 6 — Define a data structure to represent your tree and justify you choice.

2.2 Criterion

Ex. 7 — Implement the function `find_question` which is selecting the most accurate question to split a given node. The accuracy of the question is evaluated using the **entropy impurity**.

Ex. 8 — Implement the function `eval_tree` which is computing the mean of the RMSE of given corpus using a given tree.

2.3 Training the tree

To train the tree, we will process as indicated on the lecture (slide 26). However, we are going to use the previous criteria. The stopping criterion is defined as

$$eval\_tree(T_i, validation\_corpus) - eval\_tree(T_{i-1}, validation\_corpus) > \epsilon \tag{1}$$

Ex. 9 — Implement the function `train_tree` which is taking the *training* set the *validation* set and the question set as parameters. This function is supposed to train the decision tree based on the two criteria and an epsilon value. For this exercise, the epsilon ϵ value is 1.

3 Some analysis [3pts]

In order to evaluate the performance of an algorithm, we can use the k-cross validation methodology. The key idea is to generate randomly the 3 subsets; use the *training* and the *validation* corpus to train the tree. Then we use the *test* set to compute an error measure. This process is repeated k times. Then the mean of the error gives an idea of the quality of the algorithm.

In our case, we do not have any error reference to compare the achieved one. So we won't use it completely. We are still put in place the same process and analyze the results.

Ex. 10 — Propose an automatic way to randomly fill the subsets based on the original one.

Ex. 11 — Apply the decision tree clustering on 10 combinations of corpus. You should dump the tree each time. You also should generate a csv file containing 2 columns : the index of the combination and the corresponding mean RMSE value

Ex. 12 — Compare the decision trees. What can you conclude? Justify

Submission instructions

The following instructions are mandatory. If you are not following them, tutors can decide to not correct your exercise.

Submission architecture

You have to generate a **single ZIP file** respecting the following architecture:

```
tutorial1_<matriculation_nb1>_<matriculation_nb2>_<matriculation_nb3>
|
+--- source
|      |
|      +------- file 1
|      +------- file 2
|      +------- ...
+--- rapport.pdf
+--- README.txt
```

where

- **source** contains the source code of your project,
- **rapport.pdf** is the report where you present your solution with **the explanations (!)** and the plots,
- **README** which contains group member informations (name, matriculation numbers and emails) and a **clear** explanation about how to compile and run your source code

The ZIP filename has to be :

```
tutorial1_<matriculation_nb1>_<matriculation_nb2>_<matriculation_nb3>.zip
```

You have to choose between the following languages **python** or **matlab**. Other languages won't be accepted.

Some hints

We advice you to follow the following guidelines in order to avoid problems :

- Avoid building complex systems. The exercises are simple enough.
- Do not include any executables in your submission, as this will cause the e-mail server to reject it.

Grading

Send your assignment to the tutor who is responsible of your group:

- Gerrit Großmann gerritgr@gmail.com
- Sébastien Le Maguer slemaguer@coli.uni-saarland.de
- Kata Naszádi b.naszadi@gmail.com

The email subject should start with `[PSR TUTORIAL 8]`