Devices classifier

Jadwiga Świerczyńska

1 Introduction

In this report, we describe a classifier for home appliances. We are given data (from REDD dataset, `http://redd.csail.mit.edu`) on power consumption by 5 home appliances, namely: `lighting2`, `lighting5`, `lighting4`, `refrigerator` and `microwave`. The sample entries from the data are below:

`timestamp`	`lighting2`	`lighting5`	`lighting4`	`refrigerator`	`microwave`
1302930703	180	23	195	117	2
1302930721	181	23	195	119	2
1302930738	180	23	195	117	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

The `timestamp` column is the information about the time of the power consumption. It is irrelevant to us, as we will assume that subsequent power consumptions happen at successive times. The numbers in the latter columns symbolize power usage for each of the appliances in the household at a particular point in time.

Our goal is to build a classifier for these devices. To be specific, when given data in a form

`time`	`dev`
1302956021	2
1302956038	0
1302956063	0
$\vdots$	$\vdots$

where `timestamp` is the time of the power consumption and `dev` is the power usage, we want to state, what device was the most likely to produce this power consumption data. For this purpose, we will build a Hidden Markov Model (HMM) for each device from the train set. Next, for each device from the test set, we will check the likelihood that one of the 5 devices produced this power consumption data. Finally, we will choose the device that was the most likely to produce this data.

2 Hidden Markov Models

Firstly, recall the definition of the Hidden Markov Model. Let $(X_n, Y_n)_{n\in\mathbb{N}}$ be a sequence of pairs of random variables, such that $(X_n)_{n\in\mathbb{N}}$ is a Markov chain. Let $S = \{s_1, s_2, \ldots, s_N\}$ be a range of X_n for all $n \in \mathbb{N}$ and let $V = \{v_1, v_2, \ldots, v_M\}$ be a range of Y_n for all $n \in \mathbb{N}$. We call S the set of *hidden states* and V the set of *visible states*. Now let $A = [a_{ij}]_{N\times N}$ be a *transition matrix* of $(X_n)_{n\in\mathbb{N}}$, i.e.

$$a_{ij} = \mathbb{P}(X_{n+1} = s_j | X_n = s_i)$$

and let $\mu = (\mu_1, \ldots, \mu_N)$ be an *initial probability distribution* of X_1, that is $\mathbb{P}(X_1 = s_i) = \mu_i$ for $i = 1, 2, \ldots, N$. Let $B = (b_i)_{1 \le i \le N}$ be a *sequence of observation likelihoods*, i.e.

$$b_i(k) = \mathbb{P}(Y_n = v_k | X_n = s_i) \text{ for all } n \in \mathbb{N}.$$

Note that if observations are continuous, then we assume that b_i is the probability density function of observations when the hidden state is equal to v_i. If $(X_n, Y_n)_{n \in \mathbb{N}}$ is as described above, then we say that $(X_n, Y_n)_{n \in \mathbb{N}}$ is a **hidden Markov model**. We refer to the triple $\lambda = (\mu, A, B)$ also as a hidden Markov model.

There are 3 main problems regarding the HMMs:

1. Given a HMM $\lambda = (\mu, A, B)$ and a sequence of observations $O = (o_1, \ldots, o_T)$, what is the likelihood that the observations were generated by the model, $\mathbb{P}(O|\lambda)$ (or log-likelihood, $\log \mathbb{P}(O|\lambda)$)?

2. Given a HMM $\lambda = (\mu, A, B)$ and a sequence of observations $O = (o_1, \ldots, o_T)$, what is the most likely state sequence in the model that produced the observations?

3. Given a sequence of observations $O = (o_1, \ldots, o_T)$, what HMM $\lambda = (\mu, A, B)$ is the most likely to have produced these observations, i.e. what λ maximizes $\mathbb{P}(O|\lambda)$ (or log-likelihood, $\log \mathbb{P}(O|\lambda)$)?

In this project, we focus only on problems 1 and 3. The third problem is exactly about finding the model that, in a sense, approximates the test data in the best way. Hence it describes the phase of learning in our algorithm. On the other hand, the first problem is the ingredient of the process of determining which device produced given observations. Assuming that we have 5 models (one for each of the appliances), given a sequence of observations, we would like to choose the device whose model has the greatest likelihood of having produced this sequence.

The first problem is the easiest to solve. We recursively calculate $\alpha_t(i)$, likelihood of observing $o_1, \ldots, o_t$, if s_i is the hidden state at the moment t. This is the so-called Forward algorithm.

The solution to the second problem uses a similar idea. We recursively calculate $\delta_t(i)$, the likelihood that the hidden state in time t is equal to s_i, assuming that we took the most likely path in times $1, 2, \ldots, t-1$, and $\psi_t(i)$, a state in time $t-1$ maximizing the likelihood that in time t the hidden state is s_i. This is the Viterbi algorithm.

Finally, the third problem can be solved using the EM (Expectation-Maximization) algorithm.

3 Algorithm

In order to describe the algorithm, denote firstly T_1, a sequence of observations for `lighting2` from the train set, T_2 – for `lighting4`, T_3 – for `lighting5`, T_4 – for `refigerator`, T_5 – for `microwave`. Moreover denote $(O(t))_{1 \le t \le test\_size}$, a sequence of sequences of observations from the test set. Recall that our goal is to determine, which device was the most likely to produce $O(t)$ for $1 \le t \le test\_size$. The algorithm will be as follows:

1. Determine the hyperparameters of HMMs $\lambda_1, \ldots, \lambda_5$ corresponding to the 5 appliances, basing on $T_1, \ldots, T_5$ respectively.

2. For each $1 \le t \le test\_size$ calculate $a_t = \arg\max_i \log \mathbb{P}(O(t)|\lambda_i)$ and return the a_t-th device as the answer to the t-th query.

In our approach, we will use Python library `hmmlearn` (`https://hmmlearn.readthedocs.io/en/latest/`). In this library, the process of learning the hyperparameters is already implemented and the only hyperparameter we need to fix is the number of hidden states. In Section 5, we describe the process of choosing this number. Also the second step of the algorithm, i.e. finding the probability that the model produced the given sequence of observations, is implemented in this library.

4 Akaike information criterion

We will need a method for rating the models with some fixed values of hyperparameters, in our case the number of hidden states in HMM. Let O be a sequence of observations. The first approach to rating the model λ would be to look at the value of $\mathbb{P}(O|\lambda)$. However, if we set the number of hidden states in λ to the length of the sequence of observations, it would be easy to obtain a model, for which $\mathbb{P}(O|\lambda) = 1$. Simply each state would correspond to one point of time. Nevertheless, such a model probably would be an example of overfitting and would not work well on the previously unseen data. Hence we decide to use another criterion of the performance of the model, the **Akaike information criterion**:

$$aic(\lambda, O) = 2k - \log \mathbb{P}(O|\lambda),$$

where k is a number of model parameters. Assuming that we have N hidden states and observations have normal distribution on $\mathbb{R}$, then $k = (N-1) + N(N-1) + 2N$, respectively number of parameters needed to represent the initial distribution, the transition matrix, and mean and variance for observations in each state.

By using the AIC, we do not favor too complex models over the less complex ones. This way we avoid overfitting. Hence we will prefer the models that have a smaller value of AIC.

5 Number of hidden states in HMMs

We assume that the observations have a Gaussian distribution. Moreover we set `n_iter`$= 50$, a number of iterations performed by the algorithm from `hmmlearn`, and `tol`$=10^{-4}$, a convergence threshold. By `score` we mean the log-likelihood from problems 1 and 3 for HMMs and by `n_components` the number of the hidden states of the model.

Note that in this task we are given measurements from some time interval. Therefore to rate the models, we need to split the train data T_i into train and test sets, respectively R_i and E_i for $i = 1, \ldots, 5$. However, we cannot simply shuffle the set T_i and split it into two parts, as the order of the events does matter. Hence we will split the set T_i into prefix, containing $p\%$ of the T_i, and suffix, containing the rest. The prefix will be the set R_i and the suffix will be E_i. We perform experiments for $p \in \{50, 60, 70, 80, 90\}$ and present some results below. The set E_i will not be used to rate the model in order to choose the parameters. We will refer to it only to see how the model is behaving on the previously unseen data.

Fix $i \in \{1, \ldots, 5\}$. Denote hs_i, the desired number of hidden states in HMM for the i-th device. The algorithm for finding the number of hidden states in λ_i is as follows:

1. For $p \in \{50, 60, 70, 80, 90\}$:
    (a) split T_i into a prefix R_i containing $p\%$ of the data, and a suffix E_i containing the rest
    (b) For `n_components`$= 2, 3, \ldots, 20$:

i. $\lambda^p_{i,\texttt{n\_components}}$:= model trained on R_i, with random seed from $\{0,1,2\}$ (choose the one that gives the best `score` on R_i)

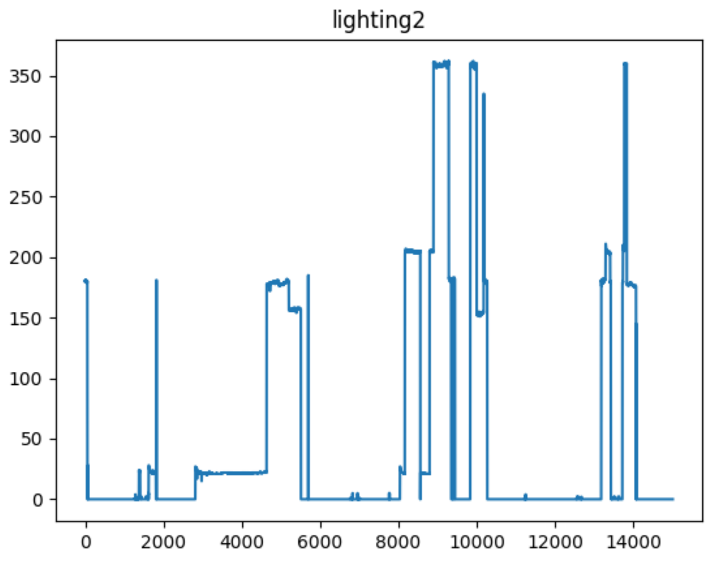
2. hs_i := the best `n_components` in terms of $aic\left(\lambda^p_{i,\texttt{n\_components}}, R_i\right)$ and $\texttt{score}\left(\lambda^p_{i,\texttt{n\_components}}, E_i\right)$ for $p \in \{50, 60, 70, 80, 90\}$

Notice that the method for choosing a random seed among a few options was suggested in the `hmmlearn` documentation (see here).

The last step is intentionally not formulated precisely. We will choose the final value of hs_i by looking at the plots of the value of AIC obtained on R_i for different values of p. The AIC provides information about the relative quality of the models (see here). However, when we consider many ways of splitting the data into R_i and E_i, then the models become incomparable, as they are trained on different datasets. Hence we will choose the value of hs_i for which values of AIC are "not so big" for all $p \in \{50, 60, 70, 80, 90\}$. We will also take into consideration how the model with the given value of hs_i performs in terms of `score` on E_i.

5.1 lighting2

Below we can see the data plot from the set T_1, representing the `lighting2`.



The plots in Figure 1 present the `score` and AIC that the model λ_1 achieved on R_1 for `n_components` between 2 and 20, for different ratios of splitting the data from T_1. The plots from Figure 2 show the `score` obtained on the test set, the first one for `n_components` between 2 and 20 and the second one for `n_components` between 2 and 10.

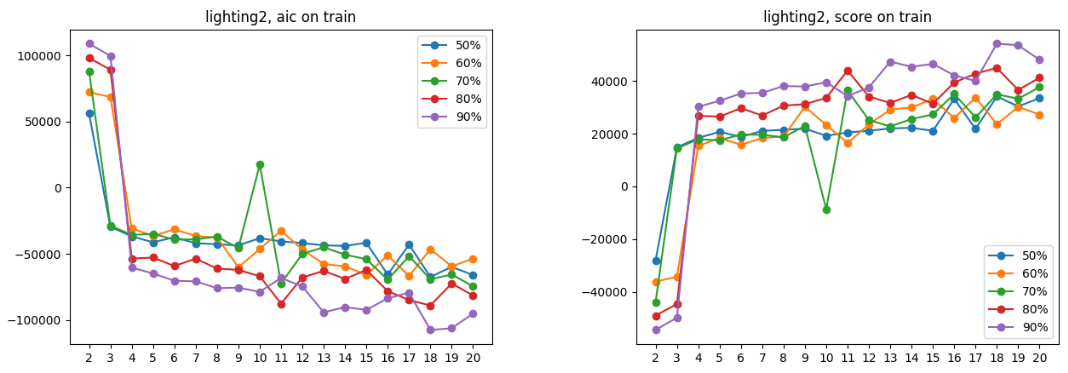


Figure 1: `lighting2` – `score` and AIC on the train set

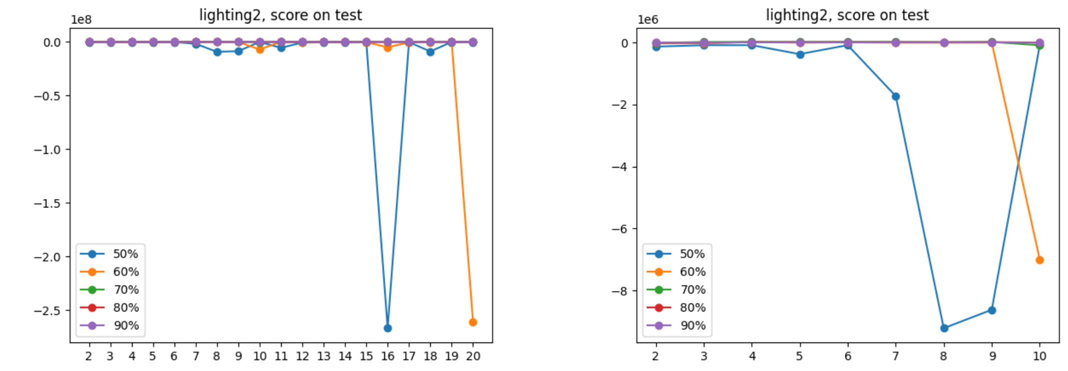
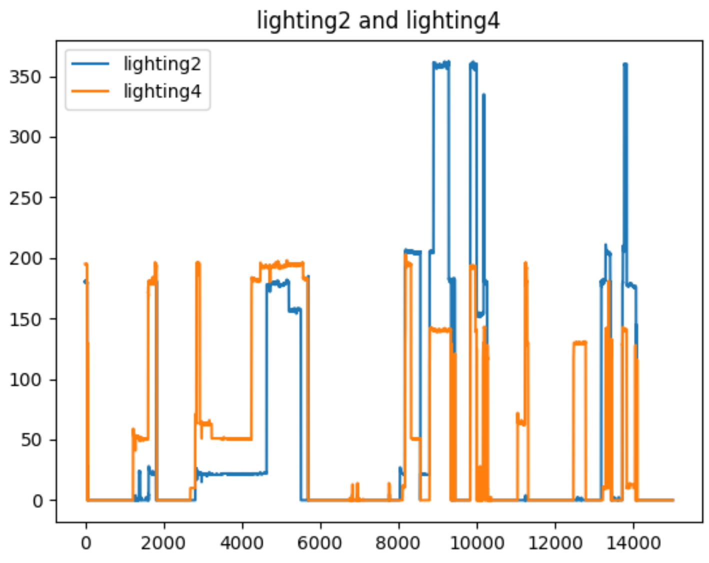


Figure 2: `lighting2` – `score` on the test set

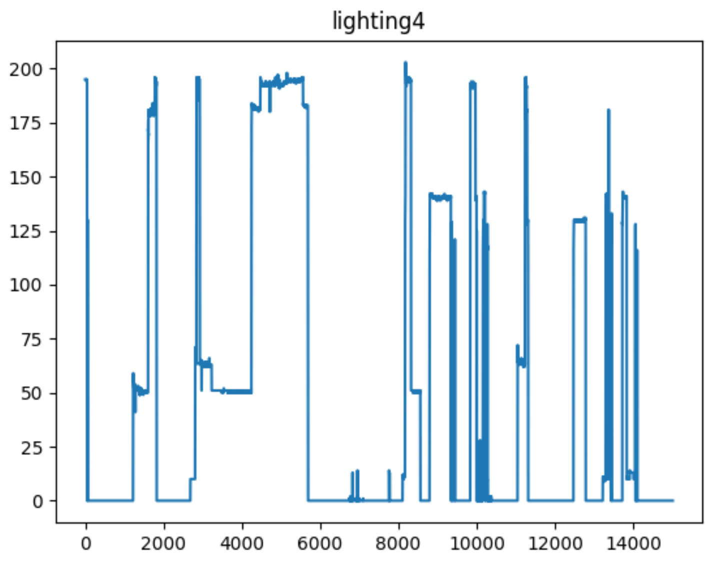
We see that big values of `n_components` may result in poor `score` on the test set; also for 7, 8, 9, and 10 there is some loss in the `score` on the test set. Here we do not see much difference between values like 4, 6 and 12, 13, 14, 15. However, we conjecture that we may need more states, as `lighting2` and `lighting4` are similar, which is presented in the figure below.



This suggests that it may be difficult to distinguish between these two appliances. Hence we prefer a greater number of states in this case and we set $hs_1 = 13$.

5.2 lighting4

First have a look at the plot of the data from T_2, representing `lighting4`.



The plots in Figure 3 present the `score` and AIC that the model λ_2 achieved on R_2 for n_components between 2 and 20, for different ratios of splitting the data from T_2. The plots from Figure 4 show the score obtained on the test set, the first one for n_components between 2 and 20 and the second one for n_components between 2 and 10.

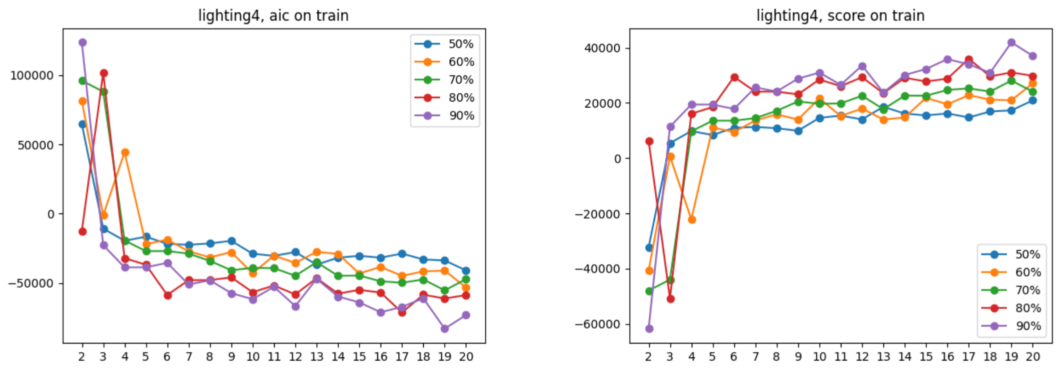


Figure 3: lighting4 – score and AIC on the train set

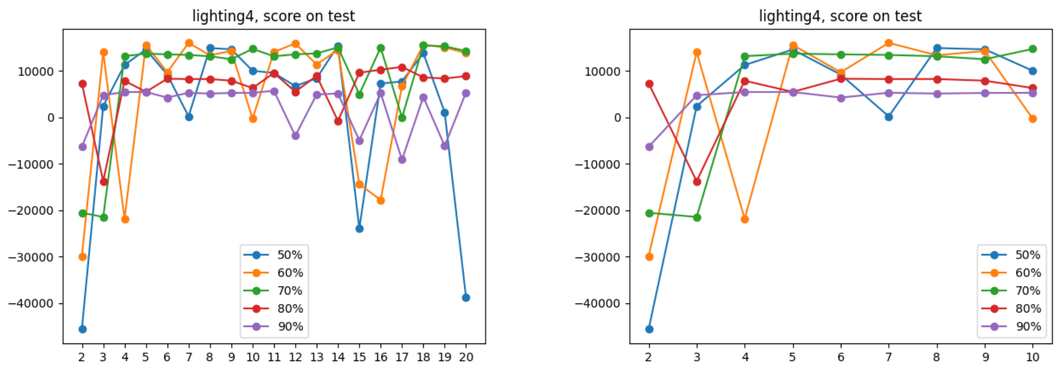
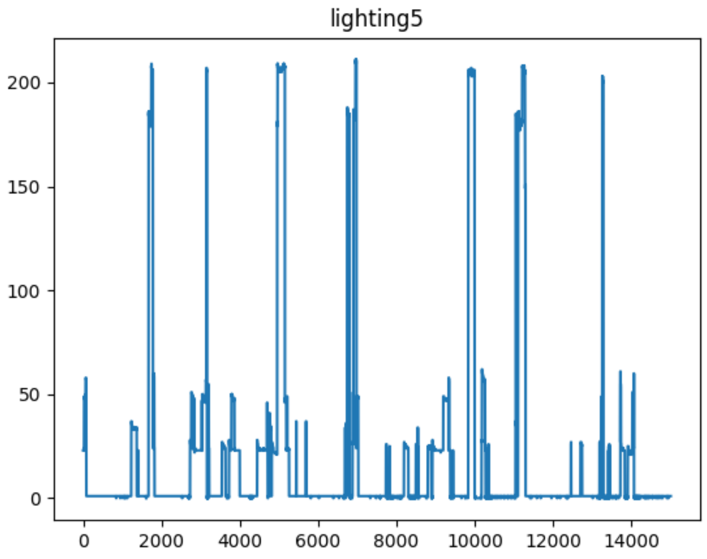


Figure 4: lighting4 – score on the test set

Here big losses in score on the test set occur for number of states greater than 14. As mentioned in the previous section, there is a chance that our models will confuse lighting2 and lighting4 as these devices are quite similar. Therefore we prefer a greater number of states for these appliances. Hence we choose $hs_2 = 14$, as both score on test and AIC on train give good results, and all of the smaller numbers of states give similar or worse results.

5.3 lighting5

Below we can see the data plot from T_3, representing lighting5.



The plots in Figure 5 present the `score` and AIC that the model λ_3 achieved on R_3 for `n_components` between 2 and 20, for different ratios of splitting the data from T_3. The plots from Figure 6 show the `score` obtained on the test set E_3, the first one for `n_components` between 2 and 20 and the second one for `n_components` between 2 and 10.

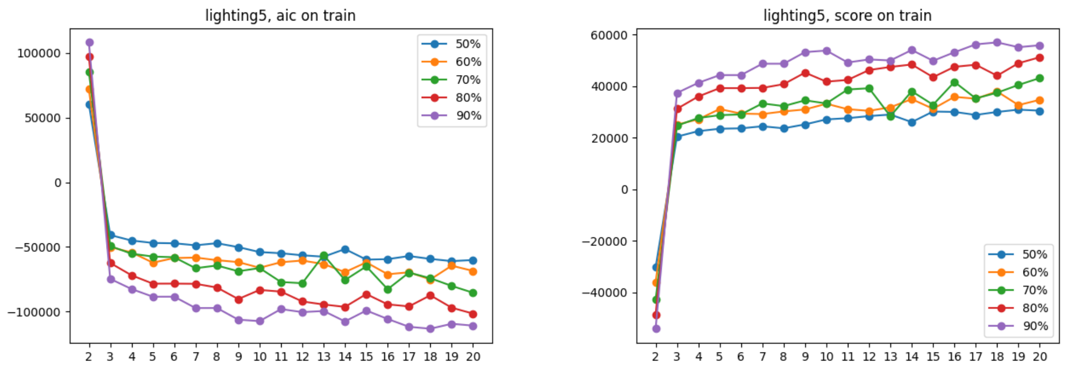
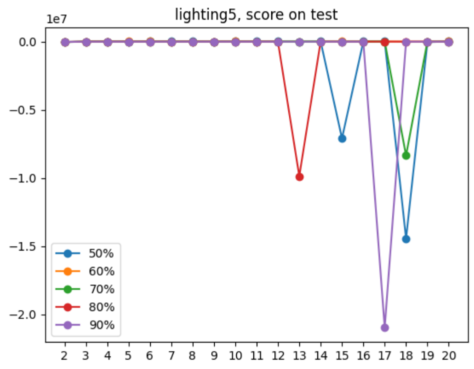


Figure 5: `lighting5` – `score` and AIC on the train set



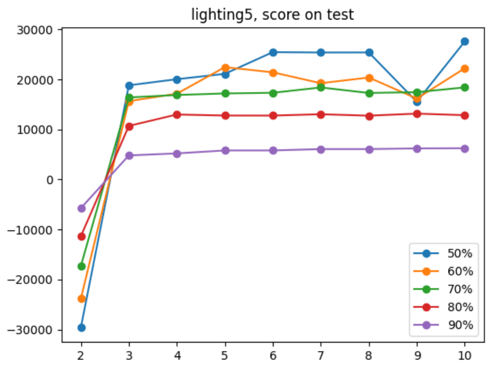
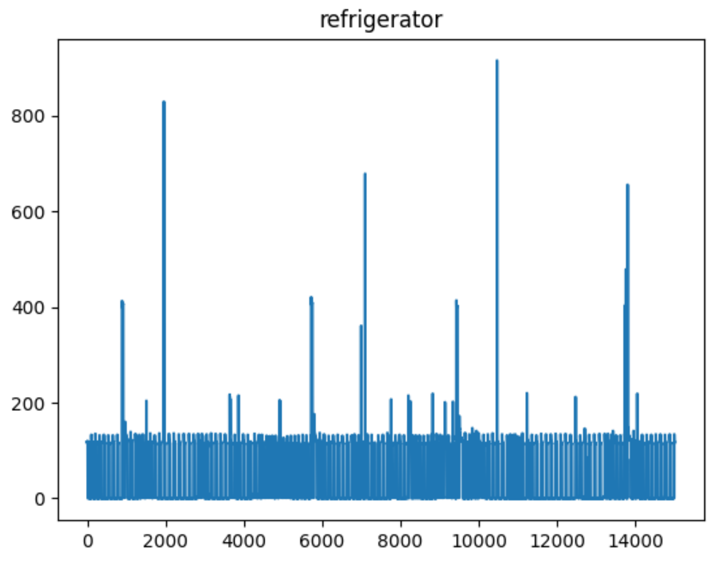


Figure 6: `lighting5` – `score` on the test set

We see that values of `n_components` greater than 12 result in some losses in terms of `score` on the test set. Hence we focus on smaller values. We see that 6, 7, and 8 give similar scores on the test set. However, in terms of AIC, 7 is the best among these three values. Therefore we fix $hs_3 = 7$.

5.4 refrigerator

Below we can see the data plot from T_4, representing `refrigerator`.



The plots in Figure 7 present the `score` and AIC that the model λ_4 achieved on R_4 for `n_components` between 2 and 20, for different ratios of splitting the data from T_4. The plots from Figure 8 show the `score` obtained on the test set E_4, the first one for `n_components` between 2 and 20 and the second one for `n_components` between 2 and 10.

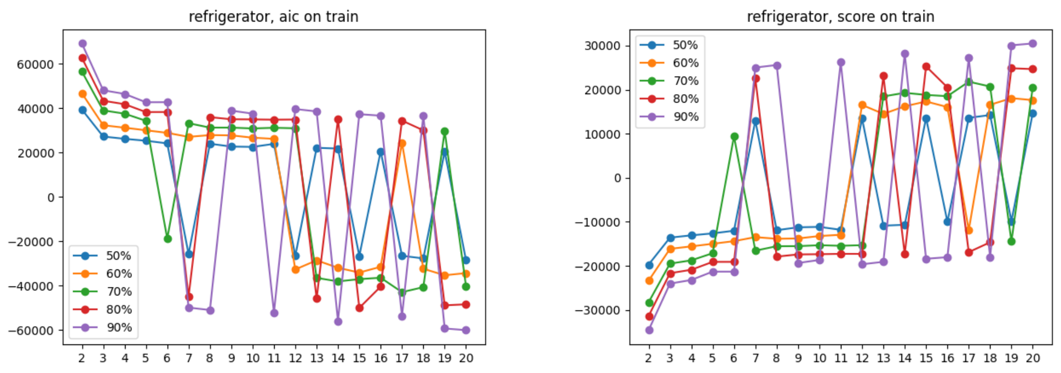


Figure 7: `refrigerator` – `score` and AIC on the train set

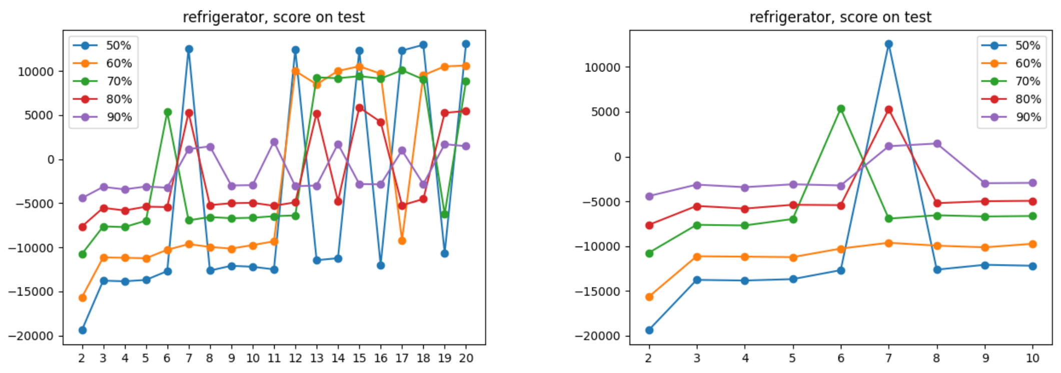
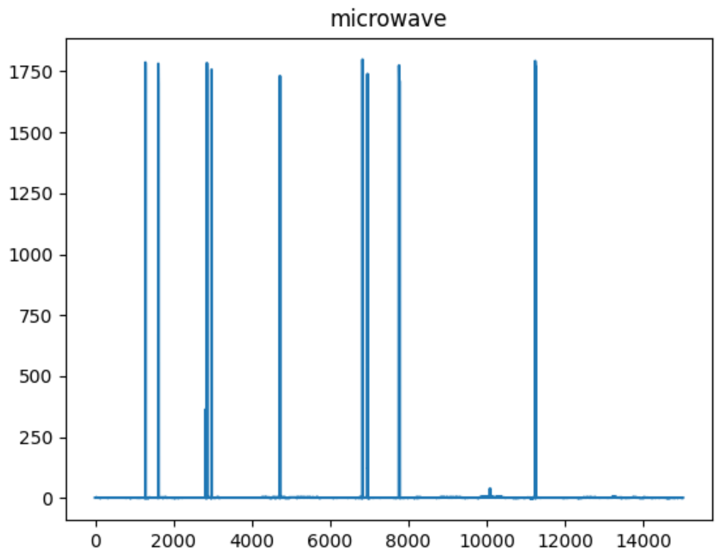


Figure 8: `refrigerator` – `score` on the test set

On the plots, we can see some periodicity for values of `score` on R_4. It is interesting as the only point at which values for all of the splits are at their peak is `n_components`=20. Moreover for `n_components`=20 the value of AIC is the smallest for all partitions. Therefore we set $hs_4 = 20$.

5.5 microwave

Below we can see the data plot from T_5, representing `microwave`.



The plots in Figure 9 present the `score` and AIC that the model λ_5 achieved on R_5 for `n_components` between 2 and 20, for different ratios of splitting the data from T_5. The plots from Figure 10 show the `score` obtained on the test set E_5, the first one for `n_components` between 2 and 20 and the second one for `n_components` between 2 and 10.

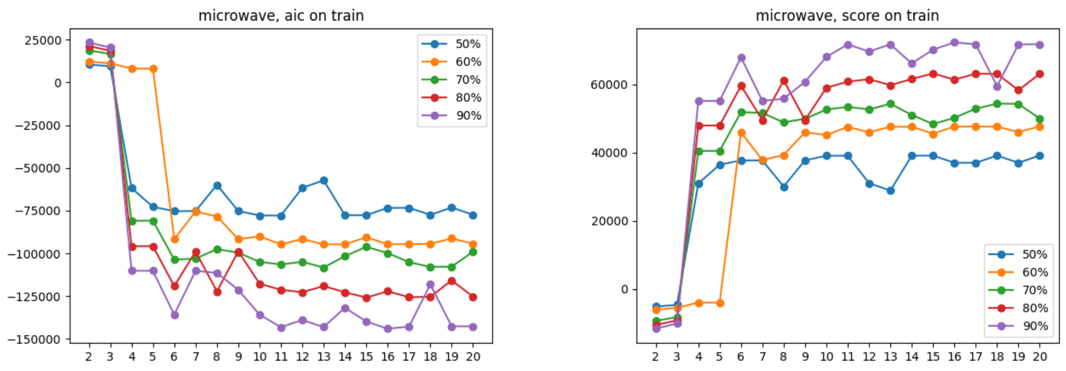


Figure 9: `microwave` – `score` and AIC on the train set

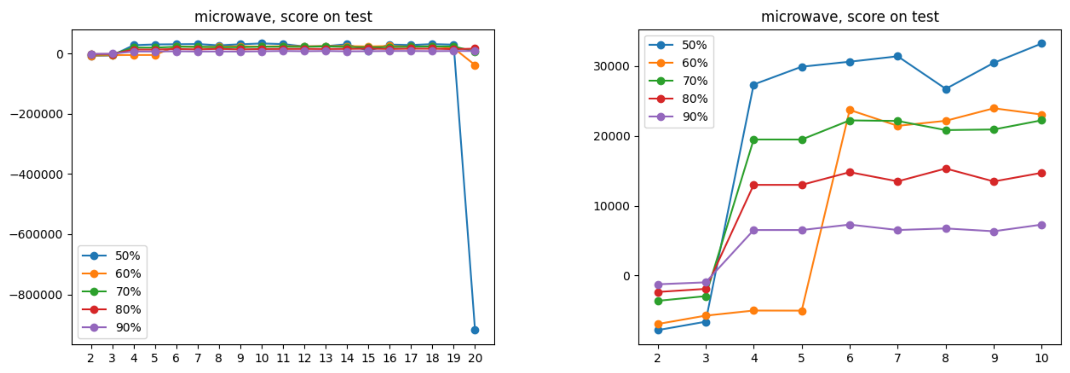


Figure 10: `microwave` – `score` on the test set

We see that the first peak of `score` on the train set (and the first pit of AIC) is in `n_components` = 6. Moreover, we see that for greater values we do not obtain significantly better results. Hence we set $hs_5 = 6$.

6 Additional tests

In this section, we will check how well our model performs when it is given fragments of the train data. Specifically, we train the models λ_i with hs_i number of hidden states on the data from T_i for $i = 1, \ldots, 5$. Then we create the test data with the following algorithm:

```
1. $t := 1$
2. For $i = 1, \ldots, 5$:
    (a) For $n = 1, \ldots, n_{tests}$:
        i. $length :=$ random integer from $[min\_length, |T_i|]$
        ii. $begin :=$ random integer from $[0, |T_i| - length]$
        iii. $test[t] := T_i[begin : begin + length]$
        iv. $t := t + 1$
```

We fix $min\_length = 1000$ and $n_{tests} = 100$. Hence we have 500 test cases (100 for each of the appliances) and we can check how the models will classify these tests. The results (number of correctly answered test cases) with a few different random seeds are below:

random seed	`lighting2`	`lighting4`	`lighting5`	`refrigerator`	`microwave`
0	100	100	100	100	100
7	100	100	100	100	100
42	100	100	100	100	100

7 Summary

Finally obtained values of the number of hidden states in HMMs for particular devices are as follows:

device	number of hidden states
`lighting2`	13
`lighting4`	14
`lighting5`	7
`refrigerator`	20
`microwave`	6

The code of a classifier can be found in the file `classify_devs_330498.py` and the scripts for performing tests can be found in the file `worker.py`.