## Summary

Hidden Markov models are usually used for extracting the hidden states of a system, where we don’t necessarily know the answer beforehand. However, it can be used in a supervised fashion to classify data into pre-known states. The best model was able to classify states in the Tarsalis dataset with an accuracy of ~90%. However, there is a possibility that the model does not converge during training.

## Preprocessing

As with UNet, the dataset is split into probes and then concat’d into a single long time series with corresponding lengths of each measurement provided. This is a requirement for the hmmlearn-library, such that it trains on a single “time”-series (invariant of actual time) but with lengths to not introduce new state transitions.

We run a train-test split with 49 train files and 13 test files that are randomly selected.

## Process

As our dataset consists of 7 different states which we wish to classify, the expected hidden states of the HMM is set to 7. The covariance is decided by the data in use and assumptions which can be made about the different states/components/features (used interchangeably):

* Full covariance: The covariance matrix for each component is full/general, with no restrictions on the placement of elements. This opens for interactions between features.
* Diagonal covariance: The covariance matrix for each component is diagonal. There is no interaction between the features.
* Tied covariance: The covariance matrix has no restrictions (full/general), but it is shared by all components.

Selecting this is crucial.

As we have a relatively small dataset but cannot assume that there is no interaction between the features, we mostly default to the tied convergence type.

## Model

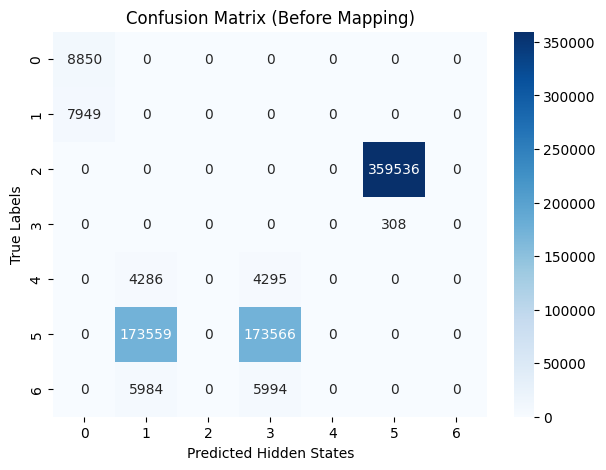
The models are trained on the post\_rect data of the whole Tarsalis dataset for 400-800 iterations, with a tied convergence and 7 expected hidden components. It is inputted as a 1-dimensional time series in addition to the lengths of the different measurements, as is required by the hmmlearn library.

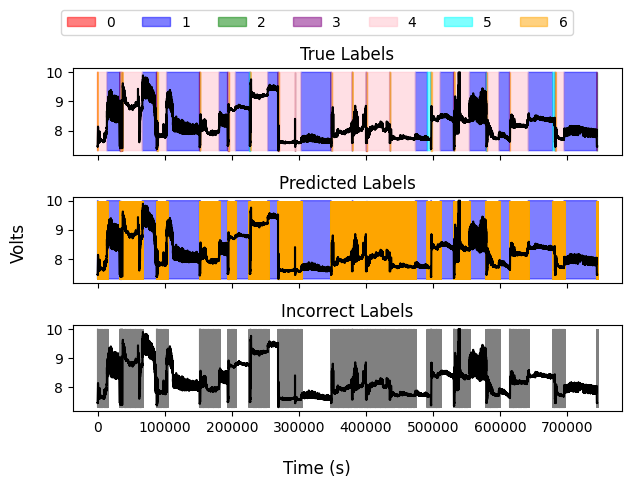
## Results

### Training on post\_rect

As the HMM is a less charted territory for classification, we first apply it to the post\_rect data as it is more well-developed. Once potential shortcomings are established on this dataset we extend it to the voltage data.

Because the hidden Markov model assigns states arbitrarily, we must do a remapping of the output states to the values of our expected states, such that they correspond to the same value. This we do via the confusion matrix, to see the strongest correlation between the output and actual states. The confusion matrix is therefore indicative, but not entirely accurate.

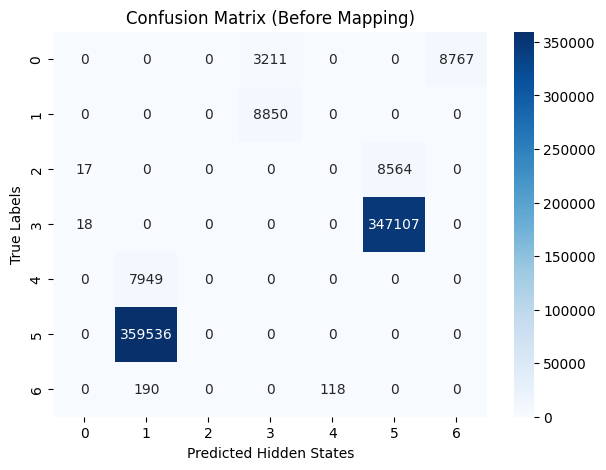




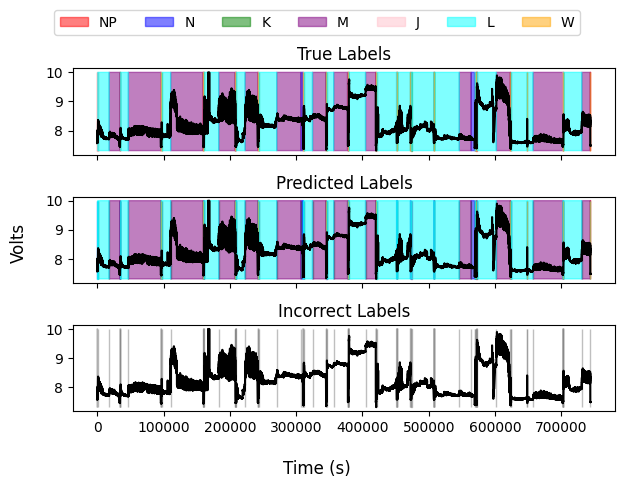
We also plot the time series overlaid with the predicted and true labels, for the entire length of the test dataset (concatenated into a single time series, as is required for the hidden Markov model). This shows that there is only a set of labels for which the HMM is accurate,

It appears that the HMM can label with a degree of confidence when there is a lot of training data. However, even in this case it is not a guarantee. In the above plots, which show a bad training output, even for the state with the most available data, the model is unable to predict it correctly. It will classify the behavior as a specific state, however that state may not correspond to the correct true state during the remapping process.

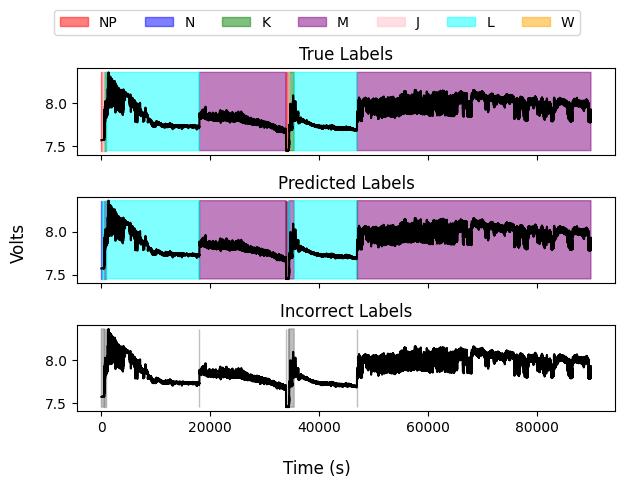
Furthermore, retraining on the same dataset with a new arbitrary state does change the model accuracy, i.e. the training is not semi-deterministic for an arbitrary initial state, as it appears the model gets stuck in local optima during training. The model presented is an example thereof. Refinement is necessary in order to make the training process somewhat deterministic (such that it does not vary between 30% accuracy and 90% accuracy).

We also present the case of better convergence below, including the confusion matrix and the true labeled time series alongside that of the predicted labels. Note again that the confusion matrix is indicative and represents the states before a remapping occurs for the hidden Markov states.

For the labeled time series, it appears that for this training run, the model struggles primarily with short-lived transitory states. In this case, however, there’s no indication of a state with which the model has a systematic problem.



A closer-up image on a specific part of the time series:



For this improved model, the performance is as follows:

Class NP: Precision = 1.00, Recall = 0.73, F1 Score = 0.85

Class N: Precision = 0.73, Recall = 1.00, F1 Score = 0.85

Class K: Precision = 0.49, Recall = 0.00, F1 Score = 0.00

Class M: Precision = 0.98, Recall = 1.00, F1 Score = 0.99

Class J: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class L: Precision = 0.98, Recall = 1.00, F1 Score = 0.99

Class W: Precision = 1.00, Recall = 0.38, F1 Score = 0.55

Overall (Macro) Statistics:

Precision = 0.74, Recall = 0.59, F1 Score = 0.60, Accuracy = 0.97

In the best case, the model performs very well on NP, M, L and W. It does very poorly on J and poorly on K.

### Training on voltage

Having established these shortcomings on a more developed dataset, we turn to the voltage data and train a model on it instead. Note that as before, the training process is non-deterministic and may change if retrained on the same data with a different initial state.

Class L: Precision = 0.93, Recall = 0.75, F1 Score = 0.83

Class J: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

Class NP: Precision = 0.04, Recall = 0.16, F1 Score = 0.06

Class K: Precision = 0.03, Recall = 0.18, F1 Score = 0.05

Class N: Precision = 0.05, Recall = 0.98, F1 Score = 0.10

Class W: Precision = 0.00, Recall = 0.00, F1 Score = 0.00

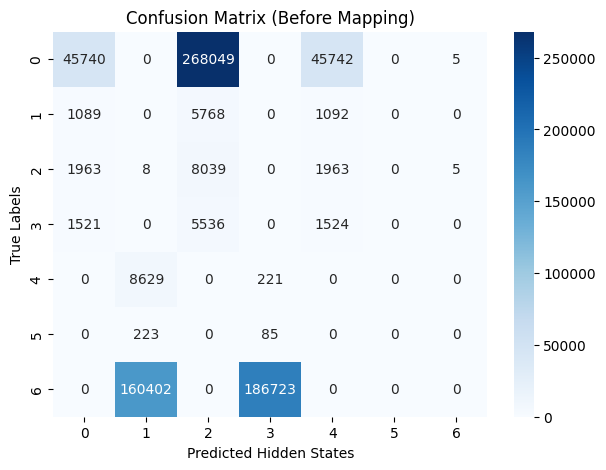
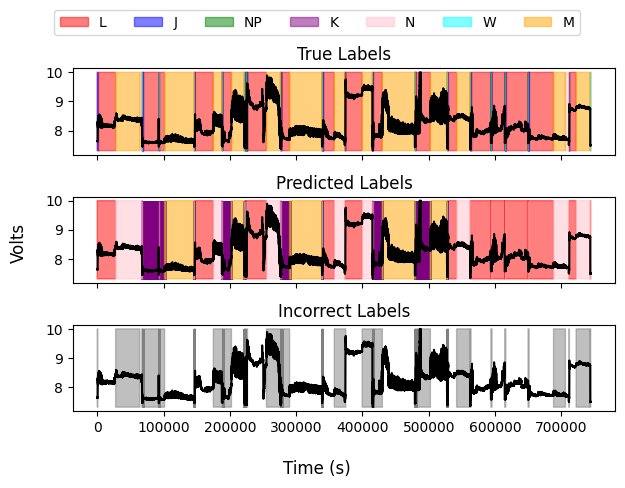
Class M: Precision = 1.00, Recall = 0.54, F1 Score = 0.70

Overall (Macro) Statistics:

Precision = 0.29, Recall = 0.37, F1 Score = 0.25, Accuracy = 0.63

The score is much lower, and the model has lost its ability to classify many of the states. It does very poorly on J, NP, K, N and W, while M and L (the most common states) are successfully classified most of the time. It is least accurate for short-lived transitory states. However, it should be noted that the states it assigns are stable and last approximately the same duration as the correct state. In short, it is able to perceive the state changes, but not classify them correctly. This is the case for all training output.

We show the confusion matrix, again before remapping the states.



## 

## 

## Next steps

The next steps shall be a refinement of the parameters such that the model can be reliably trained from a specific dataset, for all reasonable randomized initializations. There should not be as large fluctuations in the model performance as there are right now.

Beyond this, an investigation of the optimal number of parameters should be performed to maximize the accuracy.

## Conclusion

It does seem that the hidden Markov model is a viable means of classifying the states of a given dataset. However, it cannot be stated conclusively due to large variations in the quality and convergence of the outputted models for the same dataset, only dependent on the initial conditions in use. This may point to an insufficient data set.