Dissertation - Detecting User Engagement Using Mouse Tracking Data

David Saunders (910995)

September 2020

Abstract

Potential abstract. This project explores the use of Hidden Markov Models to explore users mouse cursor data while performing a task. The users are split into 2 distinct groups, online crowdsourced users, and lab study users. We propose that the lab study users were paying attention, and the crowdsourced users may or may not be paying attention. By using observed data recorded from the users mouse we are able to model both groups interaction with the system with Hidden Markov Models. Some users interaction seem misplaced compared to their groups. These are reclassified with the aim of changing the groups of user from lab study or crowdsourced, to users paying attention and not paying attention.

Contents

1 Motivation

Detecting use engagement is an important science across multiple disciplines.

Find a study that says during any task / any crowdsourced task X% of people are not paying attention. This could jeopardise the results from any online if we cannot be sure users are actually paying attention to the task they are being paid to complete.

It is a hard problem because, among other things I will touch on, it is hard to quantify exactly what user engagement is exactly. One author defined it as 'XYZ' [look at my previous work for references].

However the previous authors have failed to detect and measure user engagement as X,Y,Z.

1.1 Contributions

In this project my contributions to fix it are, -a system to classify users, a way of visualising their mouse paths, ways to directly and quantitatively compare different users, and a multistage semi-supervised based binary classification output to answer the question of 'are users engaged'.

2 Related Work

All related work about actual other attempts to detect user engagement from mouse tracking data. Other subsections will be like miniature literature reviews or something?

2.1 Semi Supervised Learning

This will be a key aspect of the project as we only have definitive labels for part of our data. This reflects the challenges of real world data, by some estimates only 2% (made up number) of all data is structured and labelled, the rest is unstructured [find reference].

In a sense the data here is labelled. All day belongs to 2 classes, online turk user, or in person lab study user. However on the other hand the data is not labelled for the task I would like to explore. The goal of this project is to identity which users were paying attention. We have assumed that we can infer that lab study users will be paying attention, so we can say that those samples are labelled. However the rest of the samples we have which are all of the online data are unlabelled. Therefore this is a kind of semi supervised learning problem where we only have a small percentage of our samples labelled, and only confident labels for one class. (See if a paper on this exists, semi supervised binary classification with only labels for one class.)

2.2 N-Grams

This paper has a nice scientific explanation of n-grams [1]. Either cite this paper or more likely look at their reference for n-grams and cite that.

2.3 Hidden Markov Models

"Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process – call it X – with unobservable ("hidden") states. HMM assumes that there is another process Y whose behaviour "depends" on X. The goal is to learn about X by observing Y." Wikipedia

In the case of this project, the hidden states X would be the users thought processes including whether or not they're paying attention. The observations, Y, are the data that has been collected from the users such as mouse location over time and sequence of interface elements.

Paper Tom send me about HMM for text classification that I might be able to use [2].

Paper claims to use HMM for spam detection, I think they actually just use it to detect misspellings of words or something which is used by spam to hide from filters [3]. Probably could find a better spam detecting HMM, I just like how this almost does something different from the title, which my diss will end up doing.

This paper was recommended by someone on stack overflow as an old influential paper in the field with tens of thousands of references [4]. Called a tutorial on HMM and its so old so original source. Would definitely be good to reference if I include any of the mathematics behind HMMs.

Someone's dissertation on the topic of generating synthetic data with HMMs. This can be a good way to create synthetic data [5].

Maybe this would belong under implementation?

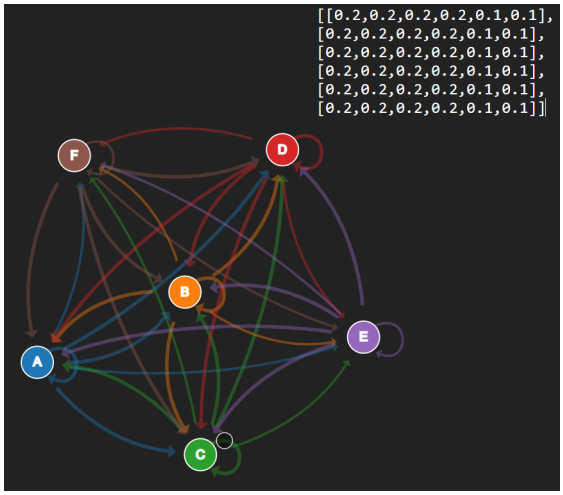

Figure 1: Diagram of a Markov Chain model representing a users mouse position

Figure 1 shows the states of a markov model of my system with states A-E representing sliders 1-5 and state F representing an html element. In the top right we can see a possible transition matrix of our system.

3 Implementation

To implement a HMM the python library HMMLearn was used [6]. The sequence data used to train the models was the order of mouse targets. The temporal aspect of the data was ignored, but earlier research showed that number of mouse events and time were heavily correlated. Therefore using just the order of mouse events should capture the temporal aspect of the data.

When implementing a Hidden Markov model on the actual data there are lots of parameters that must be considered. Given a sequence of input data the HMM model can calculate most of the parameters such as the transition matrix by using the Forwards-Backwards algorithm. The most prominent parameter that must be given is the number of hidden states to use.

The paper [7] addresses this issue on similar data. The authors use a HMM to model animal movement behaviour, where they hope the hidden states would roughly relate to the behaviour of the animal. For example a 2 state HMM on this model may relate to "foraging/resting" and "travelling". Humans can be more complex than animals but we can think of potential states of being "inquisitive" and "bored" They trained models with 2, 3, 4, and 5 hidden states and compared the criterion's of AIC, BIC, ICL. Similarity to evaluate Lab and Turk models models were trained with number of hidden states ranging from 1 to 11. Generally an increase in states should increase the effectiveness, however

it will also take longer to train. All models used for this comparison was trained to 50 iterations for a fair comparison.

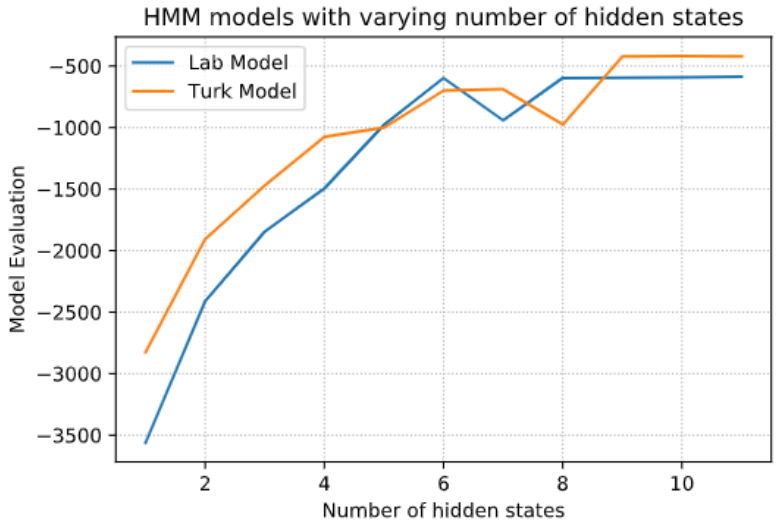

Figure 2: Line plot of evaluation of different HMM architectures.

Figure 2 shows a different criterion of the total log likelihood of the training data of the model. This value is normalised by dividing by the number of data samples, to give a per-sample total likelihood. The evaluation of each HMM model does fluctuate based on the random state used, which is why there are drops in the evaluation criterion even when more hidden states are used. We can see that the criterion for both models stop improving after 9 states. Therefore a 10 state model was used.

4 Methodology

The assumption of this project is that labs are paying attention and turks are not. Therefore if we get any outliers from the turk data, but that are similar to the lab data, then we will say that that online turk user was paying more attention than his peers, and that they were paying attention.

This study (ref) says that only 10% of all people / turk users pay attention during a task. We will look at the 10% (30ish) of the turk data that looks like it is lab data and day that they were paying attention. This is just the assumption we have made for the project, unfortunately the dataset isn't extensive enough for us to fully test this hypothesis.

5 Data Pipeline

When planning and completing this project many decisions were made about the steps taken to convert the raw data to a finished product/classification. This section may act as an overview of the project, detailing the different sections of work, what they may contain, and the order in which they will be completed.

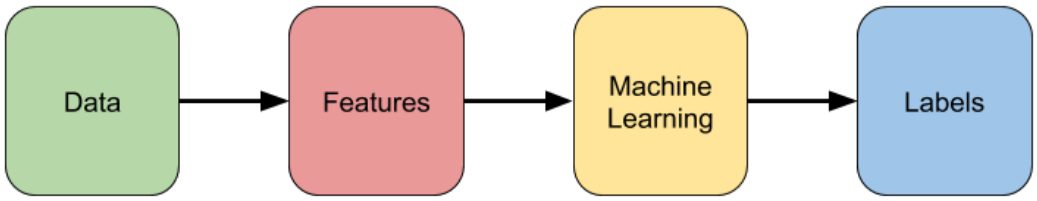

Figure 3: Diagram of the Data Pipeline of the project.

5.1 Data

Here I will summarise what I have done to the data. As previously mentioned the data was gathered through a lab study and an online study. The purpose of the online study was to gather a larger amount of data that was possible to do so in person. The aim of this was to make any results more statically significant. Data was recorded in a big JSON dump with lots of irrelevant and repealed data relating to the users background and not their mouse movements. Pythons JSON couldn't directly convert the data as mouse events were stored as a nested JSON dictionary and there were errors in the way it was written making it invalid JSON. Took ages and should explain more indepth, but then finally got data into a tabular data format of a csv which I am more comfortable working with. Ended up with over 100,000 lines? Might have been more.

This leads to the problem of imbalanced data samples. There were approximately 11 lab data and 400 online data, meaning that there were 40x as many data samples from one class compared to the other. As stated in my assumptions, we can say that the lab participants were paying attention, where as the online participants may or may not have been paying attention.

If the classes were balanced then simple approach may be to treat this problem as binary classification problem. Using something like a Support Vector Machine we could classify a given point as lab or online / paying attention or possibly paying attention based on their proximity to other data points. To do so we would need to have balanced classes otherwise the algorithm would have a high accuracy from just classifying everything as possibly paying attention as that is the most frequent class. There are two main methods of dealing with class imbalances, removing data samples and creating new data samples.

It was decided that creating new data samples would be best as there is not a whole lot of data to work with, so there would be a strong preference to keep the data we have. New points can be created by sampling from a distribution (reference) but here it was decided just to duplicate the samples as there was no discernible distribution of the data. Another method is to copy the points, altering them slightly, this was considered but not used in the end. Each lab study data sample was copied 40 times to even out the classes.

5.2 Features

This part of the pipeline refers to what features I am going to extract from the data. Features of data can be defined as 'attributes or interesting things from the data'. [reference] These will consist of both raw and created features, but what do I mean by this? Raw features will consist of the the number of mouse events recorded, while a created feature could be comparing the trace of users cursor data when using the program.

5.3 Machine Learning

Once we have insightful features from the data we can consider what machine learning algorithm would be most appropriate to use on the data. This will obviously be highly dependent on what form the final features are in. For example if the features are numerical values such as time taken to complete task and number of mouse events then an algorithm such as a Support Vector Machine would be a good choice. If the data is in the form of sequential data such as a list of all mouse events then something like a LSTM or RNN network would be best suited. If the features output was an image such as a trace of mouse position over time then a CNN could be a good choice as they're designed for image data. It is likely that text classification algorithms will be used when comparing the targets of mouse events. Comparing n-grams can be done with algorithms such as XYZ [reference]. Other text classification algorithms such as cosine similarity or sentiment analysis could also be used.

5.4 Labels

Lastly an important section of the pipeline, as it reflects the final outputs of the system. Labels will refer to which users are classified as paying attention and which users are not. A key aspect of this project will be semi-supervised learning. That is we have some data points we can confirm were paying attention, and others where they're level of attention was questionable. Once we have an algorithm that can classify some users as paying attention or not we can rerun the algorithm with these preliminary outputs as new training data. If this is done recursively then we can end up with a system that can split all data points into the 2 classes, perhaps with a degree of confidence given as a percentage.

6 Repeated Experiments

This section will give an overview of the work I have completed during this project. Each sub section offers a unique approach to tackling the problem. The different methods were tried sequentially, and the pipeline had to be amended for each one as the input requirements changed.

Approach of this project was to try many different methods of looking through, classifying, and predicting attentiveness of this data.

(Get the actual exploration and ml/data science steps by looking at notebooks.)

6.1 Distance Based Methods / Spacial Classification Methods

Not sure if this is the best name for it. These experiments attempted to separate the data samples for turkers, and lab participants by plotting samples in space. The aim is that this might reveal information about the features of different users. If there was a clear distinction between the classes then we could potentially reclassify some points based on their euclidean distance to one another.

6.1.1 SVM separation methods

Here we look at if a simple hyperplane could separate the data when looking at similar features the number of mouse events and the total time taken to do the task. It was found that a linear, or hyperbolic(wasn't hyperplane was something else) was unable to separate data. This was mainly die to the unbalanced classes.

Here simple features of the data was used, the time taken to complete the task, and the number of mouse event taken to complete the task.

6.1.2 Clustering

Didn't do this but its an example point. (Might have tried kNN).

6.2 Bag-Of-Words Methods

(Maybe mention bag-of-words models in background research?)

Can treat each target as a word as use NLP methods such as bag-of-words. With with method we will lose data regarding order but that can be analysed later with HMMs. "In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision." - ($wikipedia.org/wiki/Bag-of-words_model$)

Disregarding grammar is a big weakness of traditional uses of BoW but the concept of grammar is meaningless in the case of a sequence of mouse targets.

6.2.1 Naive Bayes

Another idea from SPAM detection is using a naive bayes, here they get the count of each word for each email and bayes it. https://towardsdatascience.com/spam-filtering-using-naive-bayes-98a341224038

Not very good, when balanced data used cross validation average of 50ish

https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/ This website has some nice references about imbalanced class distributions. (TODO: use in diss) data sampling – customized algorithms– cost sensitive algorithms– one class algorithms – threshold moving – probability calibration

$https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/$ Better ways of dealing with imbalenced data

6.2.2 Naive Bayes with N-Grams

Could even use this to get the counts of 2/3-grams to compare? More successful but suffered from same issues.

Include image of n-grams distributions? TODO: Plot of ngrams for both classes. Create side by side bar plot.

html seems to be higher with turkers rather than lab users, this may be because they were fidgeting and not paying attention?

6.3 Hidden Markov Models

HMMs are typically used with time series data. Here I just used the series of mouse events and ignored the timestamps of the event. It was found earlier that the length of mouse events and time are positively correlatied with a pearsons correlation of ?? 0.6, and so just the sequence of mouse events should still hold the detail that time data would.

These can be used for binary classification as described here (https://datascience.stackexchange.com/questio hmm-be-used-as-a-binary-classifier). Say theyre used to generate samples, maybe in another part as that will probably need more explanation.

With the bag of words model we ignored all order to the sequences looking only at the frequency of each item. This is a different approach, here the order is the most important thing.

EXPLAIN EVERYTHING IVE DONE.

6.4 Imbalanced classes

HMMs can be used for classification as shown above, but they are also known as generative models. This means they can be used to create new data. The above methods of SVM, KNN, and Naive Bayes failed due to the imbalances of classes. Now the trained HMMs could be used to generate new data points, to reduce the imbalances, and make these methods effective.

The HMM can generate a probable sequence of n observed states, however to generate new samples we must decide what length n must be.

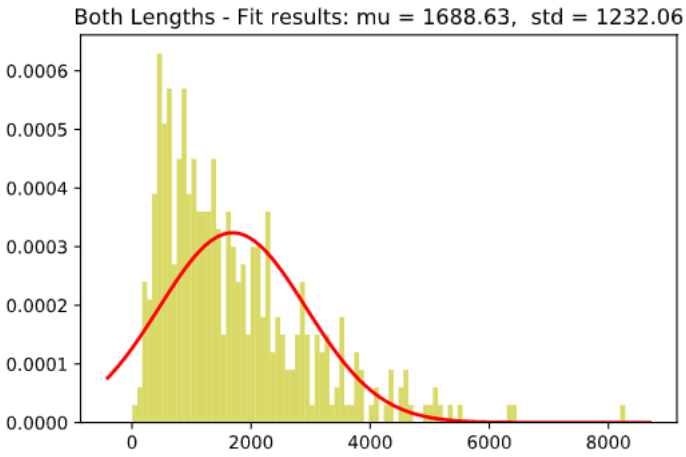

Figure 4: Histogram of length of mouse events for turker and lab data.

Figure 4 shows a the frequency of different mouse events lengths. In red we can see a positively skewed normal distribution which matches the distribution fairly well. When determining n for the new sequence data, we will sample from a gaussian distribution with a mean of 1688 and a standard deviation of 1232. This should generate a realistic distribution of lengths, and when combined with the trained HMM model, a realistic sequence of data.

6.4.1 Revisiting methods

With the generated lab study data we can now revisited some of the methods that failed due to imbalanced classes. With the naive bayes of counts of different mouse targets, there was little improvement, with an accuracy and f1 score of 49% and 46% respectively. However there was a big improvement when the bi-grams of events were fed into the model. The accuracy and f1 scores increased to 94%, a dramatic increase. This shows that there is enough difference in the data for a classification algorithm to distinguish the points.

7 Results

Now the real aim of the project can begin, attempting to classify users as paying attention or not. We will take the 2 HMMs and feed it in the existing mouse data. This will tell use the likelihood that that sequence of mouse events belongs to each class. If there are any points that appear to be outliers, then we can say that they were doing something differently to the majority of the turkers. If the same outlier also has a higher likelihood of belonging to the HMM trained on lab data then we can say that that outlier sample appears to belong to the

wrong group. Or at least that outlier is doing something different to their peers, which may be a higher level of attention given to the task.

7.1 Table or graph of results

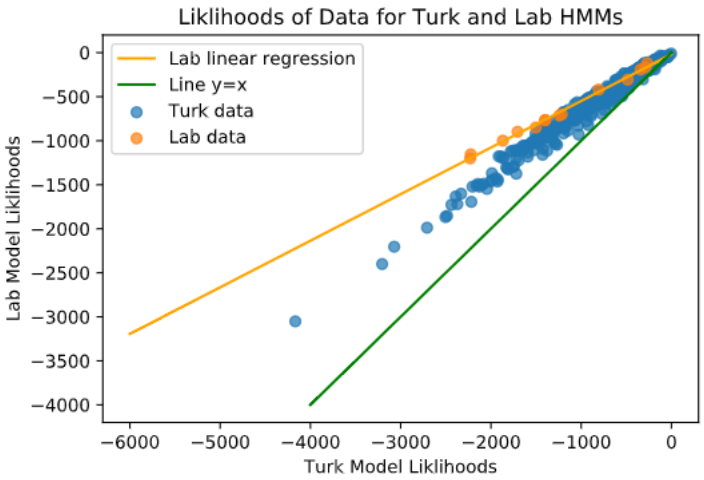

Figure 5: The comparative likelihoods of users classifiers with the Turk and Lab data.

Figure 5 shows how closely each users mouse sequences match the different HMM models.

The model likelihoods are the the log likelihoods that each user belongs to the lab or turk models. One way of attempting to classify users is to identify any users with a higher lab likelihood as being more similar to a lab user than a turk user , however all the points have a higher lab likelihood than turk likelihood (shown by the green line y=x).

An alternative solution is to plot a regression line of the lab users likelihoods (shown in orange). We can see that the lab data points form an almost straight line. Any points above this line have a higher likelihood of being a lab user or a lower likelihood of being a turk user than the actual lab data. Thus this regression line forms a sort of pareto frontier, with points above it a more feasible choice of lab-like data than points beneath it.

Figure 6 shows which of the turk users are more similar to the lab users , than other turk users. This plot shows the same datapoints and axis , but the points above the line are recoloured yellow to show their difference from the actual lab data (green) and normal turk data (purple). 120 of the 461 (26%) turk users are above this threshold. Which seems like a reasonable percentage of the population to be paying attention. The exact distribution of these points change based on the initial random state when training the model.

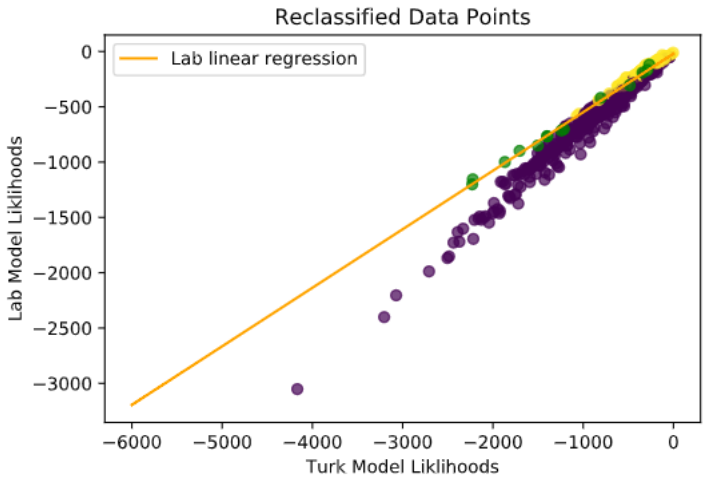

Figure 6: Plot showing the turk users that have been reclassified as behaving more like lab users.

7.2 Likelihood length correlation

Figure 7

8 Conclusion

To evaluate the results I would not be too confident in my findings. The data was incorrectly labelled for the task I wanted to perform. Additionally the data was heavily imbalanced. All of the conclusions I have made from the lab data are relying on only a handful (13 maybe) of datapoints which is not statistically significant. Additionally there was not a lot of data from the other class of online data either. Typically data science and machine learning uses big data, where as this project used only 400ish records in total which came to only 500mb of data (MAYBE). Any bias in any of these original samples will be magnified as this was used to classify more points, which would again spread this bias. This would become a self fulfilling prophecy as more similar points would be labelled and spread the belief.

Biggest flaw of the project is the first assumption, that the lab people were all paying attention. While they were monitored and we can be sure they were not distracted by phones or televisions, many of them may have 'zoned out' and not have been given it their full effort and attention.

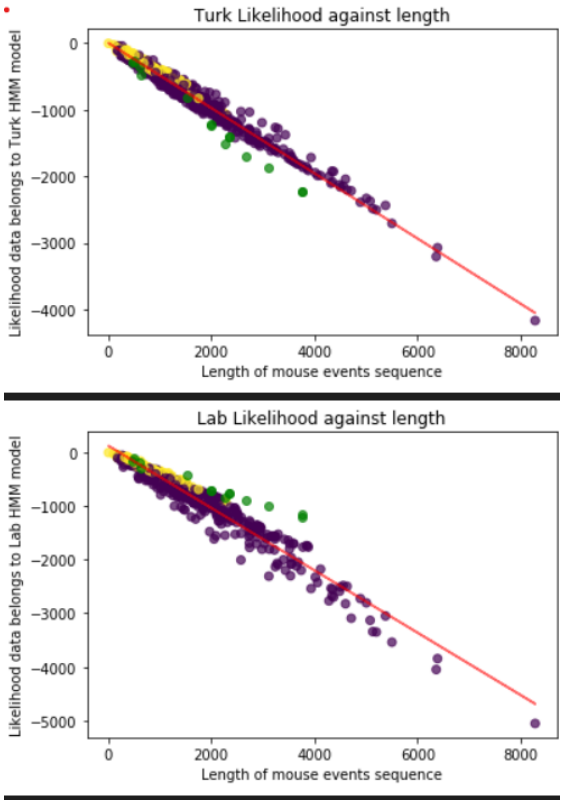

Figure 7: Plots showing correlation between likelihood of different mosules and teh length of mouse events.

9 Future work

It would be interesting to see if any of the methods here would reveal anything interesting in other datasets. A kaggle crowdflower dataset seems like an idea candidate and I would be interested if we could determine if any of these users were paying attention.

References

[1] Andrija Tomović, Predrag Janičić and Vlado Kešelj. "n-Gram-based classification and unsupervised hierarchical clustering of genome sequences". In: *Computer methods and programs in biomedicine* 81.2 (2006), pp. 137–153.

[2] Michael Collins. "Tagging with Hidden Markov Models". In: (2016).

[3] José Gordillo and Eduardo Conde. "An HMM for detecting spam mail". In: *Expert systems with applications* 33.3 (2007), pp. 667–682.

[4] Lawrence R Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[5] Jaime Ferrando Huertas. *Generating synthetic data through Hidden Markov Models*. 2018.

[6] Hmmlearn developers. *hmmlearn*. `https://github.com/hmmlearn/hmmlearn`. 2020.

[7] Jennifer Pohle et al. "Selecting the number of states in hidden Markov models: pragmatic solutions illustrated using animal movement". In: *Journal of Agricultural, Biological and Environmental Statistics* 22.3 (2017), pp. 270–293.