Radboud University

Faculty of Social Sciences

Bachelor Artificial Intelligence

Academic year 2016-2017

Date: 18 June 2017

How generative models develop in predictive processing

Bachelor's Thesis Artificial Intelligence

Author:

S.H. Aalbers

4454537

S.Aalbers@student.ru.nl

Department of Artificial Intelligence

Radboud University Nijmegen

Under supervision of:

Dr. J.H.P. Kwisthout

Abstract

In the predictive processing theory generative models are used to make predictions about future inputs, only processing the unpredicted parts: the prediction error. In order to be able to make increasingly better predictions, predictions errors should be used to update and revise the generative models. How this model generation and revision is established is still unclear. This research focuses on comparing a basic- and a more specified generative model to find out if this is a possible way that the generative models develop. This research will be done using Lego Mindstorms EV3-robots.

Introduction

The theory of predictive processing is a theory assuming that the brain tries to predict the inputs that it will receive. It assumes that instead of processing all the inputs it will receive, the brain only processes the part of the inputs that it did not predict. This is a theory that summarizes the entire operation of the brain (Hohwy, 2013). The brain predicts its inputs by using (stochastic) generative models that describe the stochastic relations between causes and effects.

It is still unclear how these generative models are made and updated. Maaike ter Borg researched a possible manner of how a first generative model is made. She researched if it was possible to make a generative model based on k-means clustering. She concluded that the method isn't very useful for motor commands which are initiated by the researcher because it is dependent of the initiated commands, but that the clustering method is very useful for the sensory input.

In general, a person gets better in things when it has done it more and more often, this is called learning. In terms of predictive processing this means that the generative models are getting better and better because the prediction errors are decreasing and the predictions are getting more informative. This means that the models are developing over time, but as yet it is not well understood how generative models can develop. In this project we investigated how a most basic generative model (consisting of two binary variables, one for the causes and one for the effects, each based on two clusters) can become more detailed. In particular we investigate the effect of re-clustering into three clusters and the effect on the generative models that this introduced, since this potentially introduced interaction effects.

Theory

The predictions in predictive processing are driven by generative models. Generative models are probability distributions that compute the tries to predict the its new inputs. For tossing a fair coin this probability distribution is: P (head) = 0.5, P (tail) = 0.5. In generative models there is a trade-off between making predictions that are very likely to be correct because they are so general, and thus carry little relevant information and predictions that will give much information because of their specificity, but are likely to be incorrect. (Kwisthout, Bekkering, & van Rooij, 2017). Predicting whether the outcome of a fair dice is odd or even is relative easy, the probabilities are both 0.5. The information you have about the outcome of the dice is not very much if you only know that it is odd or even, because then there are still 3 possible numbers left that can be the outcome. Predicting whether the outcome of a fair dice is 1,2,3,4,5 or 6 is a harder task, all the probabilities are 1/6. The information of this prediction is much higher.

We assume that the generative model starts at a general model and when it learns more, it makes it predictions more detailed (Kwisthout, 2016). For example let's assume we have a biased dice, which chances of outcomes 1,2,3,4 and 5 are 1/12 and the chance of the outcome 6 is 7/12. The generative model could start with predicting in general if the outcome will be odd or even. Its distribution will be 0.5 for both outcomes. After several trials it would notice that even occurs more than odd, so it might update its predictions to P (even) = 0.6 and P (odd) = 0.4. By doing more trials it will notice that the new predictions are good (i.e. that prediction error does not decrease any more after subsequent trials), so it can further specify its predictions. Now it can predict the chances of outcomes 2, 4 and 6 and check which of them occurs the most.

The performance of a generative model is not only expressed by its prediction error, it is also important to take its level of detail in account. The level of detail of a generative model is defined as the

state space granularity (Kwisthout, 2013) of the distribution, better known as the number of categories in a probability distribution. The uncertainty of a prediction is defined as the entropy (Shannon, 1948) of the corresponding probability distribution. When you divide the entropy of a distribution by the log of its state space granularity, you get the relative Shannon redundancy (Shannon, 1948) which normalizes the amount of uncertainty of a prediction independent of the amount of categories in the prediction.

Since we assume that generative models develop from general predictions to more precise predictions, we want to know how they develop. We are going to research this by comparing the performances of a simple generative model with a more precise model. These models are based on Bayesian models, as shown in *figure 1*.

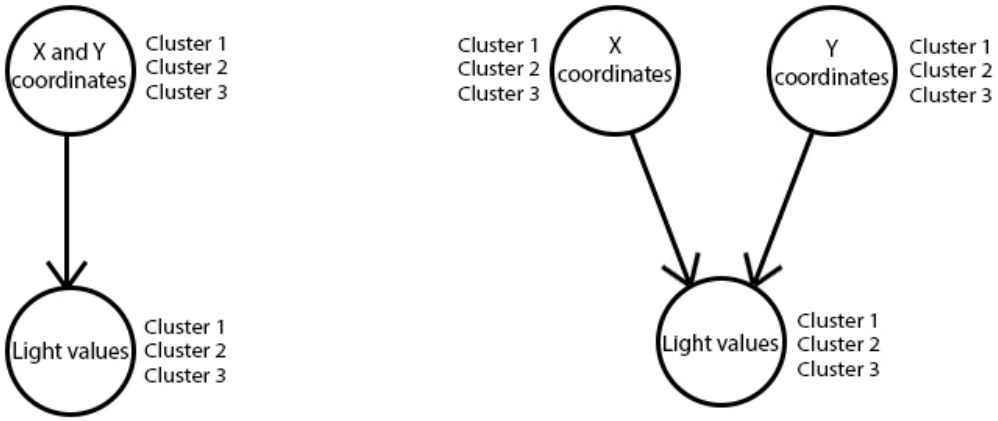

Figure 1 – Two different Bayesian models.

In the left model the x and y coordinates together are clustered into three clusters which each predict what the probability is to end up in one of the three light values clusters, this is the basic model. In the right model the x and y coordinates predict apart from each other what the probability is to end up in one of the three light values clusters, this is the precise model.

The reason we want to research these two models is because we assume that for a new situation a generative model first makes it predictions over all the variables combined. When its

predictions are good enough we think that it will switch to a more precise model in which it first predicts over all the variables apart and then combines those predictions.

This research will be done using a robot. According to the robo-havioral methodology Otworowska et al. (2015) using robots can find gaps and ambiguities in neuroscientific theories. Instead of using computer simulations we rather use robots because they force us to take reality into account. How generative models develop is currently mostly a verbal theory about how the brain operates in reality, and not a mathematic theory which can be researched in simulations.

Methods

To answer our research question we wanted to do an experiment with at least two causal variables and at least one depending variable. We have chosen for an experiment with a robot in a two dimensional field. In one corner there is a light source, the robot started in the opposite corner and its goal is to catch light at different coordinates in the field. The generative models are trying to predict for a specific combination of the causal variables what the value of the depending variable will be. The two causal variables in this experiment are the two axes of the field. The depending variable is the sum of the two light sensors.

The environment

This experiment is done on the floor of the Robotlab in the Spinoza building of the Radboud University. The whole room was dark and the only light in the room came from the lamp which was the light source. Because there was a maximum X and Y value, there was no need to mark the field in which the robot could drive. The lamp was exactly at the maximum coordinates. The light almost reached till the starting point of the robot. *Figure 2* shows an illustration of the experiment and *figure 3* shows a picture of it.

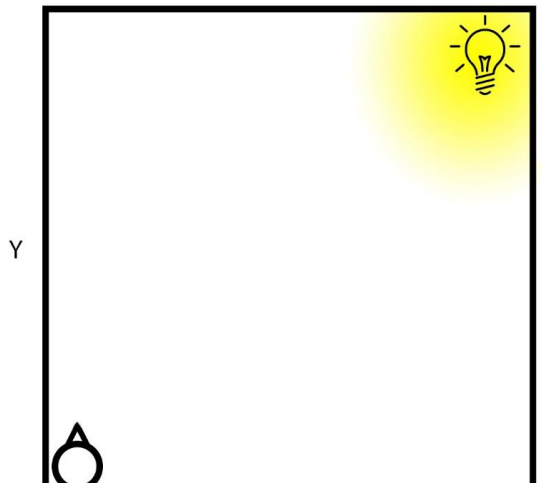

Figure 2 – An illustration of the experiment with the robot in the lower left corner and the light source in the upper right corner.

The robot

Figure 3 – A picture of the experiment. The robot is in the lower left corner and the light source in the upper right corner.

For this experiment we used a Lego Mindstorms-EV3 robot. To make it possible for the robot to drive around in a two dimensional field we constructed it using two motors. We attached one motor to each side of the robot so it could drive forward, backward and could turn around. When the robot had to drive to a coordinate, it first drove straight forward until it reached its y coordinate and then turned right and drove until it reached its x coordinate. After reaching its coordinates it stops and starts measuring the light. The robot always ended up in a state where the light source was straight before the robot (when Y was maximal), straight left to the robot (when X was maximal) or some state in between.

To make it possible for the robot to catch light we gave the robot two light sensors, one attached to the front of the robot and one attached to the left side of the robot. Since the light source is always to the front, to the left or in between of the robot, the robot always would catch at least some light. The robot is shown in *figure 4*.

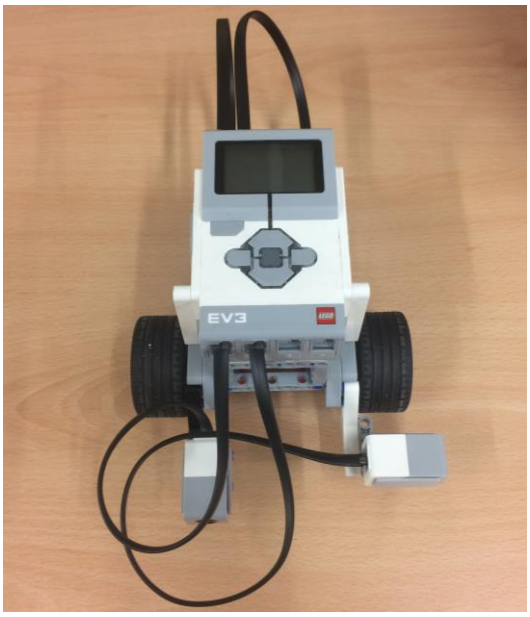

Figure 4 – The robot.

Experiments

In order to let the generative models create predictions there first should be some data they can use to predict. We gathered this data by doing forty-four trials in which we generated a random X and Y coordinate and let the robot drive to it and measure the light from there. For every trial we collected the coordinates and the values of the both light sensors and computed the sum of the values of the two light sensors. These sums of the light sensors values represents the total light intensity measured from that coordinate. These total light intensities are the data we clustered into three clusters using k-mean clustering. The joint probability distribution of ending up in one of these clusters is the goal which the generative models are going to predict. After we have all the predictions we will compute the relative Shannon redundancy. In our experiments we investigate which generative model structure results in the lowest average prediction error and the lowest average relative Shannon redundancy.

The first generative model predicts the light intensities based on de combination of the X and Y coordinates. We clustered the X and Y coordinates together into three clusters and computed the corresponding predictions from these clusters to the light intensity clusters. We computed these predictions as follows: for example for the first cluster of the coordinates we counted how much trials ended up in light intensity coordinate 1, 2 and 3. For each of these sums we divided this amount by the total number of trials in the first cluster of the coordinates. So if the first cluster of the coordinates exists of 10 trials, and only one of these trials ended up in light intensity cluster 1, than the prediction of ending up in light cluster one when your motor coordinates are in motor cluster 1 is 1/10 = 0.1. We did the same computations for the other two light intensity clusters.

The second generative model predicts the light intensities based on the combination of the clusters of the X coordinates and the clusters of the Y coordinates. First we clustered the X coordinates into three clusters, second we clustered the Y coordinates into three clusters and then we combined these into nine new clusters. We made these clusters by taking the nine intersections between the three

x-axis clusters and the three y-axis clusters as new clusters. This method is illustrated in *figure 5*, the left and middle figures represent the y- and x-axis's clusters and the right figure represents the nine combined clusters. After we made our nine clusters we could compute the predictions for these clusters. We did this in the same way as we did for the three motor clusters.

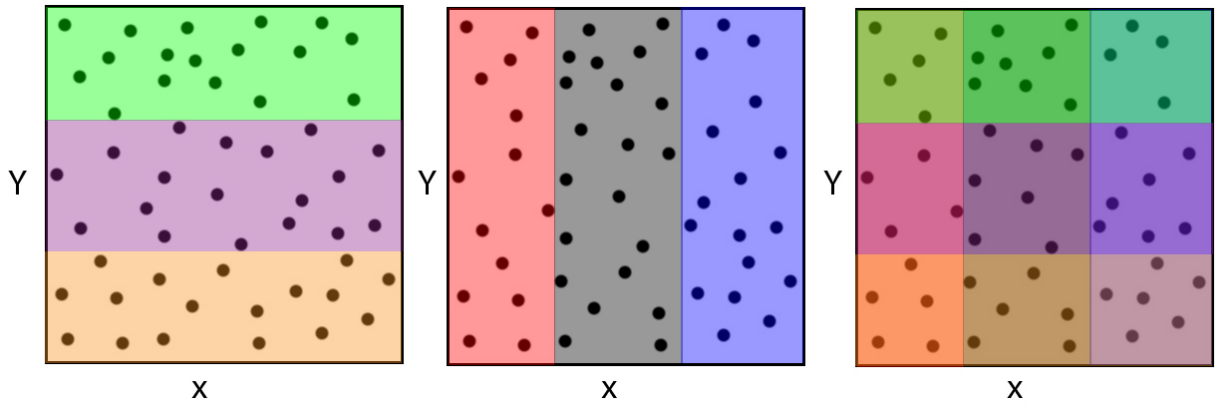

Figure 5 – How we created the nine clusters of generative model 2. Every color corresponds to a different cluster.

Subsequently we tested how good the predictions of the two generative models are. We tested this by computing predictions for both generative models for a dataset of sixty data points. These predictions were made in the same way we did as we did for the first sets of predictions, but instead of creating new clusters we used the same locations of the clusters we used in the first set of predictions.

After all the predictions were known, we were able to compute the prediction errors. We did this by taking the absolute difference between the first predictions and the second predictions of the generative models. Finally we computed the average over the two prediction errors to be able to compare them.

The final part of the experiments was computing the relative Shannon redundancies. We computed this for every probability distribution we have had in this research. To compute them first the

entropy was needed for every probability distribution, this can be computed by using the formula: $H(Pred) = \sum_{(x \in Pred)} P(x) \log_2 P(x)$ (Kwisthout et al, 2016). Now we only had to divide the entropies by the log of the number of probabilities of the corresponding probability distribution. Next we averaged all the relative Shannon redundancies that corresponded to the same generative model during the same wave of predictions, so we finally ended up with four average relative Shannon redundancies.

Clustering

All the clusters that we made are made with use of the k-means cluster algorithm. K-means clusters n data points into k clusters. It works as follows: first it places k random data points, which are called the means. Then it measures for every data point from the dataset which mean is the nearest, and then assigns itself to the cluster of that nearest mean. The next step is that the center of the clusters will be chosen as the new means (this time it actually are the means of the clusters). These three steps repeats itself until convergence has been reached. I used MATLAB to analyze my results and cluster my data. The k-means algorithm is build-in in MATLAB, and is set to k-means++ by default. K-means++ is an improved version of the k-means algorithm, which is both faster and more accurate than the original k-means algorithm (Arthur & Vassilvitskii 2007).

Results

The experiment started with gathering data from random coordinates. The sensor values are plotted in *figure 6.* As you can see the sensor values never both got much light together, it was always much light for one of the two or none. The reason for this is that the sensors are located with an angle of 90 degree between them, they don't have an overlapping field. The random data points were used to make the first clusters, the light intensity clusters, these are also shown in *figure 6*.

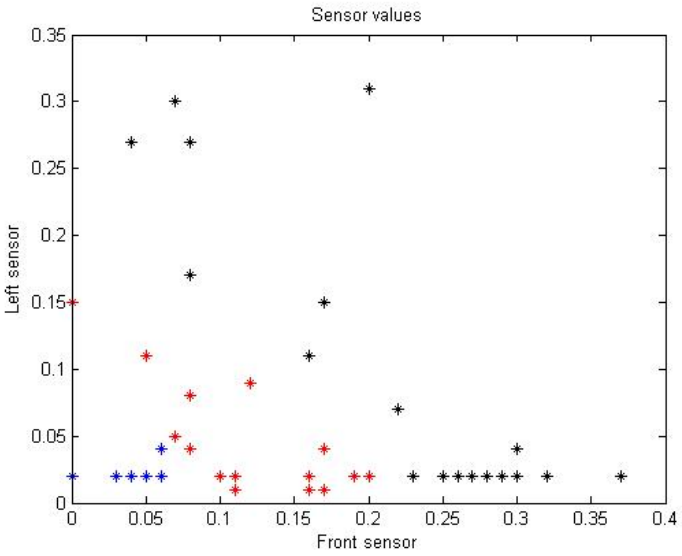

Figure 6 –The light values and clusters of the random generated data.

The data of the motor coordinates is presented in *figure 7*. As you can see the random coordinates are equally distributed over the square field. After the random data was gathered, it was time to create the generative models. The first generative model, the one clustered on the combination of the X and Y coordinate, is shown in *figure 8*.

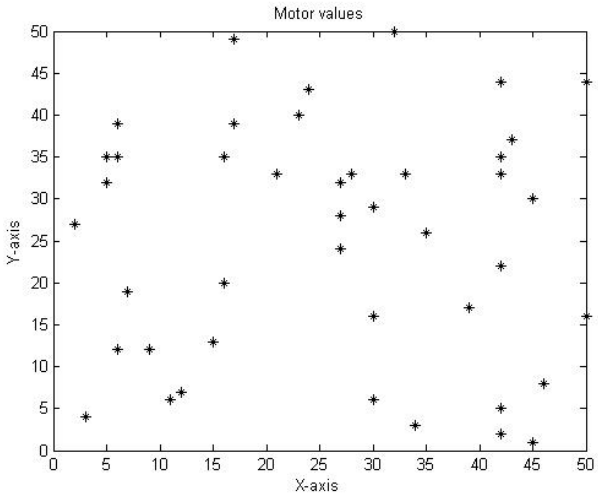

Figure 7 – The random generated coordinates.

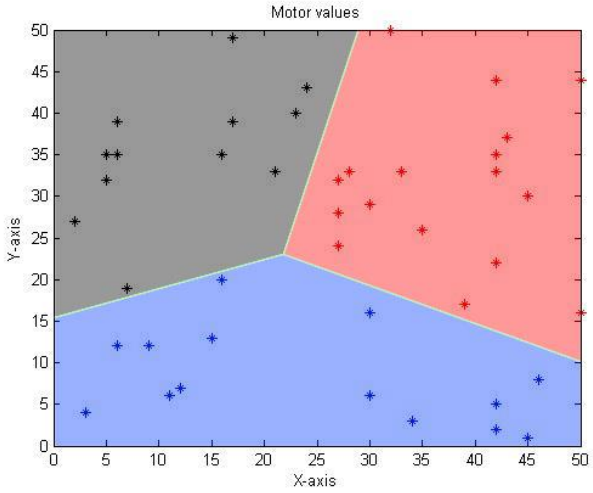

Figure 8 – The clusters of generative model 1. Each color corresponds to a different cluster: cluster 1 is blue, cluster 2 is black and cluster 3 is red.

For the second generative model a different manner of clustering was used. First the X and Y axis are clustered per axis, and those clusters were combined to new clusters. In *figure 9* and *figure 10* the clusters per axis are shown. The combined clusters are shown in *figure 11*.

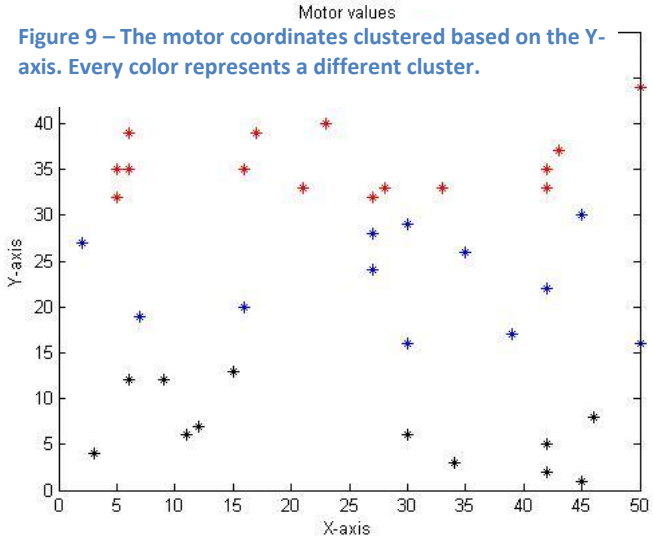

Figure 9 – The motor coordinates clustered based on the Y-axis. Every color represents a different cluster.

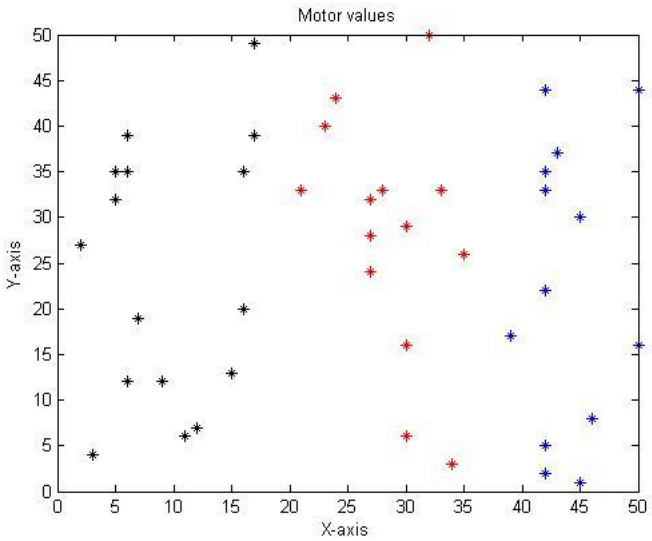

Figure 10 – The motor coordinates clustered based on the X –axis. Every color represents a different cluster.

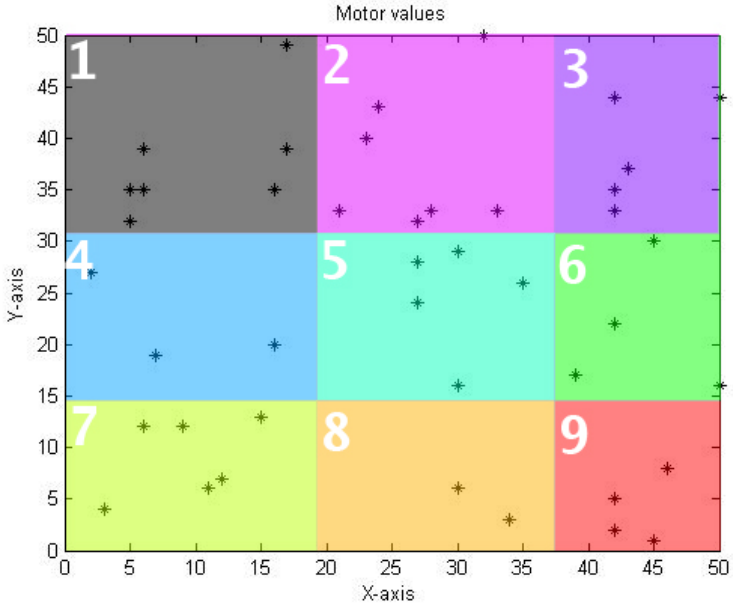

Figure 11 – The clusters of generative model 2. Every combination of color and number represent a cluster.

After the clusters where made, the predictions could be calculated. The predictions of the first and the second motor clusters are presented in *table 1* and *table 2* respectively.

Motor cluster	Sensor cluster	Probability
1	1	0.294
	2	0.353
	3	0.353
2	1	0.091
	2	0.364
	3	0.545
3	1	0.063
	2	0.625
	3	0.312

Table 1 – The probability distributions of the first predictions of generative model 1.

Motor cluster	Sensor cluster	Probability
1	1	0.143
	2	0.428
	3	0.429
2	1	0
	2	0.571
	3	0.429
3	1	0
	2	0.6
	3	0.4
4	1	0
	2	0
	3	1
5	1	0.2
	2	0.4
	3	0.4
6	1	0
	2	0.75
	3	0.25
7	1	0.833
	2	0
	3	0.167
8	1	0

	2	1
	3	0
9	1	0
	2	0.75
	3	0.25

Table 2 – The probability distributions of the first predictions of generative model 2.

To able to test which of the two generative models gives the best predictions there first should be second predictions. For the two generative models we computed the probability distributions for the sixty new data points. The data points are shown in *figure 12* and the probability distributions are shown in *table 3* and *4*.

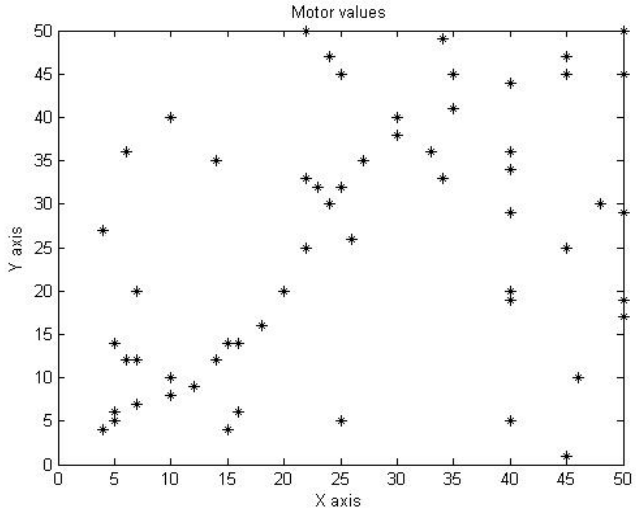

Figure 12 – The motor values of the second set of data points

cluster	cluster	Probability
1	1	0.522
	2	0.304
	3	0.174
2	1	0.273
	2	0.454
	3	0.273
3	1	0.039
	2	0.423
	3	0.538

Table 3 – The probability distributions of the second predictions of generative model 1.

Motor cluster	Sensor cluster	Probability
1	1	0
	2	0.667

	3	0.333
2	1	0
	2	0.571
	3	0.429
3	1	0
	2	0.286
	3	0.714
4	1	1
	2	0
	3	0
5	1	0.75
	2	0
	3	0.25
6	1	0
	2	0.5
	3	0.5
7	1	0.588
	2	0.294
	3	0.118
8	1	0
	2	0
	3	1
9	1	0
	2	0.667
	3	0.333

Table 4 – The probability distributions of the second predictions of generative model 2.

Now all the predictions are computed we could finally compute the average prediction errors.

Average prediction error generative model 1: 0.161

Average prediction error generative model 2: 0.276

Next it was time to compute the second part of the results of the generative models. We did this by first computing all the entropies, and then divide those by the state space granularity. The results are presented in *table 5, 6, 7* and *8.*

Motor cluster	Entropy	Relative Shannon redundancy
1	1.58	0.997

2	1.322	0.834
3	1.12	0.707

Table 5 – The entropies and relative Shannon redundancies for the first predictions of generative model 1.

Motor cluster	Entropy	Relative Shannon redundancy
1	1.449	0.914
2	0.985	0.621
3	0.971	0.613
4	0	0
5	0.152	0.096
6	0.811	0.512
7	0.651	0.411
8	0	0
9	0.811	0.512

Table 6 – The entropies and relative Shannon redundancies for the first predictions of generative model 2.

Motor cluster	Entropy	Relative Shannon redundancy
1	1.451	0.915
2	1.54	0.972
3	1.189	0.750

Table 7 – The entropies and relative Shannon redundancies for the second predictions of generative model 1.

Motor cluster	Entropy	Relative Shannon redundancy
1	0.918	0.579
2	0.985	0.621
3	0.863	0.544
4	0	0
5	0.811	0.512
6	1	0.631
7	1.334	0.842
8	0	0
9	0.918	0.579

Table 8 – The entropies and relative Shannon redundancies for the second predictions of generative model 2.

Finally we could compute the average relative Shannon redundancies.

Average relative Shannon redundancy for generative model 1 its first predictions: 0.846

Average relative Shannon redundancy for generative model 2 its first predictions: 0.409

Average relative Shannon redundancy for generative model 1 its seconds predictions: 0.879

Average relative Shannon redundancy for generative model 2 its seconds predictions: 0.479

Discussion

During the building of the robot we found out that when we wanted to robot drive straight forward it drove straightforward but when it stopped it turned a little bit to the right. We reasoned that this was caused by stopping the right motor before the left motor in the program. We fixed this by making the robot turn exact the same angle that it turned right to the left.

In the results we saw that the light values for both the light sensors together never were very high, it was often only one of the two light sensors which got a high value or none of the light sensors got a high value. This is no problem for this research if it still gives a consistent representation of how much light the robot captures at a certain location, which is the goal that the generative models try to predict. To make sure that the value light the robot captured at a certain location was constant we compared the light sensor values for different trials to the same coordinates and found that the light intensity was almost identical. The light sensor values of coordinates that were very close located to each other were also very close to each other.

The second dataset, the one of sixty data points, had to be of that size because now it would be used to compute predictions for nine clusters, instead of only for three clusters. These sixty data points were not as random as the first forty-four data points, because we used the data of earlier sets of trials which we eventually didn't need anymore. We didn't need that data anymore because we made some errors in designing our experiment which caused superfluous data. We think that these sixty data points were divided uniform enough to give sufficient data points for every cluster.

Conclusion

By looking at the results we see that the first generative model has a lower average prediction error (0.161) than the second generative model has (0.276). This doesn't have to mean that the first generative model also performs better than the second generative model does. We still have to take the average relative Shannon redundancies in account. When we look at those, we see that they are higher for the first generative model (0.846 and 0.879) than that they are for the second generative model (0.409 and 0.479). This means that the second generative model gives better matching predictions.

The goal of the research was to find out if we could find a basic generative model and a possible follow up generative model. We found a model for which the follow up model was more precise, but the predictions didn't improve, so the follow up model didn't overall perform better than the basic model. We think that using larger datasets can result in better predictions for both of the models and even result in better predictions for the more specified model than the basic model.

There are a lot of possibilities for further research to move on with this part of finding out how generative models develop. First of all the same experiment can be repeated using much more trials resulting in more robust predictions. In addition there can be used more clusters and/or different types of Bayesian models.

References

Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.

Ter Borg, M. (2017). *How generative models are created in Predictive Processing*. (unpublished bachelor's thesis)

Hohwy, J. (2013). *The predictive mind*. Oxford University Press.

Kwisthout, J. (2013). *Most inforbable explanations: Finding explanations in Bayesian networks that are both probable and informative*. In L.C. van der Gaag (Ed.): Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer Lecture Notes in AI 7958, pp. 328-339.

Kwisthout, J. (2016). ERC Starting Grant 2016 Research proposal

Kwisthout, J., Bekkering, H., & van Rooij, I. (2017). To be precise, the details don't matter: on predictive processing, precision, and level of detail of predictions. *Brain and cognition*, *112*, 84-91.

Otworowska, M., Riemens, J., Kamphuisa, C., Wolferta, P., Vuurpijla, L., & Kwisthout, J. (2015). The Robo-havioral Methodology: Developing Neuroscience Theories with FOES. In *Proceedings of the 27th Benelux Conference on AI (BNAIC'15)*.

Shannon, C.R. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal, 27*, 379–423.