What Can We Learn from Absent Cues?

Master's Project Thesis

Sanne Poelstra, s2901560, s.poelstra.1@student.rug.nl
Supervisors: Dr. J. van Rij-Tange & Dr. J. Nixon

Abstract: (250 words, placeholder) Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarksgrove right at the coast of the Semantics, a large language ocean. A small river named Duden flows by their place and supplies it with the necessary regelialia. It is a paradisematic country, in which roasted parts of sentences fly into your mouth.

Even the all-powerful Pointing has no control about the blind texts it is an almost unorthographic life One day however a small line of blind text by the name of Lorem Ipsum decided to leave for the far World of Grammar. The Big Oxmox advised her not to do so, because there were thousands of bad Commas, wild Question Marks and devious Semikoli, but the Little Blind Text didn't listen. She packed her seven versalia, put her initial into the belt and made herself on the way. When she reached the first hills of the Italic Mountains, she had a last view back on the skyline of her hometown Bookmarksgrove, the headline of Alphabet Village and the subline of her own road, the Line Lane. Pityful a rethoric question ran over her cheek, then she continued her way. On her way she met a copy. The copy warned the Little Blind Text, that where it came from it would have been rewritten a thousand times and everything that was left from its origin would be the word "and" and the Little Blind Text should

1 Introduction

Imagine that you are the first pirate space explorer to set foot on a new planet. Everything here is new and you have no idea what all these different shaped objects are. All you know is that you are going to search for treasure. How will you learn all these new concepts and if they will lead to treasure or not? When we first come into this world all input is new. So then how do we learn these new concepts?

Some literature, as will be discussed below, suggests that people learn through a process called Error Driven Learning (EDL). In this theory the main goal when learning something, is minimising the uncertainty about upcoming states in the world. EDL has been shown to work in multiple domains of learning, such as language acquisition (Hsu et al., 2011; St. Clair et al., 2009, e.g.), second language learning (Ellis, 2006). And in fields such as social psychology, category learning and more as well (see Siegel and Allan (1996) for a general review,), which means that it is not constrained to explaining only one part of learning.

To give a concrete example of this, let's say you have a smoke detector in your house and you see the light blink on it. You are not quite sure what it means, but the next day you hear it beep like it does when the batteries are empty. There is now a connection between the light blinking, the cue, and the batteries of the detector being empty, the outcome. However on that same day as you saw the lights blink just as you got out of the shower and the hallway was a bit steamy as well. So there is also a connection between the light and the shower steam. There is still uncertainty. The next time the lights blink, you do not come out of the shower and the next day the empty battery beep sounds. Now the connection between the light and the shower steam is weakened, while the connection between the light and the detector's batteries is strengthened. This way uncertainty about the world is reduced.

This is a practical example of what EDL is. However there are different proposals on how the exact mechanisms behind EDL should work. In this research we will disentangle two of those mechanisms, to hopefully gain more insight into happens when learning new concepts.

The first of the two models that attempt to describe the underlying mechanisms of Error Driven Learning, is the Rescorla-Wagner model (1972). The (simplified) formalisation of this model can be seen in Equation 1.1. ΔV_{ij}^{t} is the change in weights between cue i and outcome j at a certain time and η is the learning rate (typically set to 0.01). In the original paper η is replaced by α and β, where the first is the learning rate for the cue

and the second the learning rate for the outcome. However the current form is more similar to the Delta formula (Widrow & Hoff, 1960), and is chosen this way to limit the amount of parameters to fine-tune when modelling this process. act_j^t is the activation of outcome j, which is the sum of the connection weights for the present cue.

$$\Delta V_{ij}^t = \begin{cases} 0 & \text{, cue } i \text{ absent,} \\ \eta(1 - act_j^t) & \text{, cue } i \text{ and outcome } j \text{ are present,} \\ \eta(0 - act_j^t) & \text{, cue } i \text{ present but outcome } j \text{ absent} \end{cases} \tag{1.1}$$

This formula states that if a cue does not appear, then there is no change to the connection between the cue and outcome. So if you do not see the light blink, regardless of whether the smoke detector beeps the next day or not, there is no change to the connection. This is shown in the first line of the equation. The second line states that if a cue appears and the outcome appears as well, the connection between that cue and outcome is strengthened. If the cue appears but the outcome does not, then the connection between the two is weakened (see line three).

The reason they do not change the weights if the cue is not present, is that Rescorla and Wagner focus on the informativeness of a cue. If a cue does not appear, then that cue is therefore not informative about a certain outcome and there should be no updates to the weights.

Error Driven Learning has some other important characteristics too that we will discuss here. One of the main features of EDL is that it does not only work by association, but also by disassociation. We can see how this works in the last part of Equation 1.1, when there is a cue but the outcome does not show up. Here the connection weight of the cue to the outcome would go down. This is to ensure that the weights are updated in respect to the whole network. This would ideally result in the model learning what *not* to expect. If there is only one outcome occurring at a time, then this would ideally result in only one outcome being expected at a point in time, while all others are discarded.

However in the example we said that if we do not have shower steam and we do have a blinking light then this cue and outcome would also disassociate and their connection becomes weaker. This in contrast to the Rescorla-Wagner model stating that if a cue does not appear (shower steam) then there is no update to the weights. So how does this lead to a lower connection? This happens in a process called cue competition. All of the weights to the outcomes should add up to one together, this means that there are limited resources and the more cues are seen, the fewer those resources are. Each of the cues are competing for informativity and therefore the resources diminish as the learning process continues on. This leads to a process described by Kamin (1967) as the blocking effect. If one outcome is already highly predicted by other cues, and the resources are already divided, then a new predictive cue will not develop the same connection strength as the rest.

This also leads into another aspect of EDL, namely that it takes each time step or each trial as a separate step, so temporal order is important. This is why the new predictive cue introduced later has more difficulty getting to the same level of connection, unless it is proven to be more predictive over a long period of time. Such an aspect is for example less important in statistical learning where input is calculated in batches and the temporal order is not taken into account in theory.

In the same vein as cue competition, EDL also has outcome competition. Until now in our example there have been multiple cues, but not multiple outcomes. If we do have multiple outcomes but only very few cues, it is difficult to fully reduce uncertainty, as one cue cannot fully predict multiple outcomes. Therefore the goal of outcome competition is to get the expectation of a certain outcome given a cue.

All of these things lead to the fact that an EDL network is asymmetric. Learning does not lead to two-way associations between cues and outcomes. Rather, cues predict outcomes, but not vice versa.

As already mentioned, Rescorla and Wagner said that if a cue is not present, the cue is not informative and therefore there is no update to the weights. Several researchers did not agree with this, for example Markman (1989), who stated that we often actively encode certain features as missing. Tassoni (1995) also stated that an absence of a stimulus could just be information about the correlation between that stimulus (or cue) and outcome as well. One of the main theories that agrees with this stance of a missing cue being informative (and even based their model on that of Markman), is that of Van Hamme and Wasserman (1994), whose model is the second model of Error Drive Learning that we will discuss in this paper.

Van Hamme and Wasserman proposed that accounting for absent cues is something that *is* necessary in describing Error Driven Learning. While they agree with Rescorla and Wagner on the second and third line of Equation 1.1, they introduce an addition to the equation.

If we look at our smoke detector example from before, not seeing the light but either hearing or not hearing the smoke detector go off, should be treated differently according to Van Hamme and Wasserman. The formalisation of their theory

can be seen in Equation 1.2. While they, just as Rescorla and Wagner, give an α and β as the learning rates for the cues and outcomes respectively, here we simplified that again into one learning rate of η.

Firstly they propose a decrease in strength in connection when the cue is not there but the outcome is, as can be seen in the first line. They actively encode a missing cue as *absent* and do this by setting the learning rate η_1 to -0.01 instead of its normal positive value of η_2. Secondly there is an increase in connection strength when neither cue nor outcome is present, as can be seen in the second line of Equation 1.2. In this line the value of the learning rate is negative as well. So while the first and second line look like they are the same as the third and fourth, they will lead to opposite connection strengths.

$$\Delta V_{ij}^t = \begin{cases} \eta_1(1 - act_j^t) & \text{, cue } i \text{ absent but outcome } j \text{ present,} \\ \eta_1(0 - act_j^t) & \text{, cue } i \text{ and outcome } j \text{ absent,} \\ \eta_2(1 - act_j^t) & \text{, cue } i \text{ and outcome } j \text{ are present,} \\ \eta_2(0 - act_j^t) & \text{, cue } i \text{ present but outcome } j \text{ absent} \end{cases} \tag{1.2}$$

Before we continue discussing how Van Hamme and Wasserman tested their model, it is first important to understand that Error Driven Learning might be an implicit process. Ramscar et al. (2013) looked at how adults and children learn in the context of Error Driven Learning. They found that children showed different results in a word learning task than adults, which they attributed to adults applying reasoning strategies whereas children only seemed to use implicit learning. The reason for that is that the prefrontal cortex, the part of the brain responsible for processes such as logic or reasoning (explicit inference), is not fully developed in children yet. The development of the prefrontal cortex might be responsible for the difference in results between children and adults, as it allows adults to use explicit task strategies, while children cannot use those yet. Children's results looked more similar to what EDL predicts would happen, while the adults diverted from that prediction.

Implicit learning, according to Reber (1989), is best done without interference from logic and reasoning. Therefore explicit inference might interfere with EDL, if EDL is indeed an implicit process.

Van Hamme and Wasserman performed an experiment in the same paper they proposed the adjustment in to see if it predicted behaviour correctly. In this experiment, participants had to determine if certain foods resulted in an allergic reaction or not. They had 48 participants divided over 6 groups and each participant saw 3 sheets. One sheet consisted of 3 types of food A, B and X (e.g. peanuts, shrimp and yogurt respectively). X was always seen and A and B both 50% of the time. This means that participants saw either A and X together or B and X together in any given trial. Whether the foods lead to an allergic reaction or not depended on the condition. After each trial, of which there were 16 per sheet, Van Hamme and Wasserman explicitly asked participants for ratings. These ratings reflected how likely participants thought the three food items were to result in an allergic reaction or not. They were especially interested in the rating of the food that was not present in the trial, so either A in a BX trial or B in an AX trial. The ratings given were on a scale from 0 to 8, with 0 indicating that these foods were very unlikely to cause an allergic reaction, 4 was neutral and 8 was very likely. Participants were given fifteen seconds to look at the cues, outcomes and to fill in their ratings. After the 16 trials they were not asked for an overall score.

Van Hamme and Wasserman found that if a certain cue is present, the ratings increase over time when there is an outcome present and decrease an outcome is not present. This is both in line with what they expected and with what the Rescorla-Wagner model predicts. If there is no cue present and there is an outcome, then the ratings decreased over time. If there was no cue and no outcome, then the ratings increased. This is in line with the addition to the Rescorla-Wagner model that Van Hamme and Wasserman suggest.

Van Hamme and Wasserman only tested their new theory on human participants however, and they did not run simulations of their predictions. Therefore they did not have any formal predictions about the difference between their model and that of the Rescorla and Wagner model. One of the aims of the current study is the simulate the experiment and generate formal predictions of how both models would behave.

Another thing Van Hamme and Wasserman did in their experiment was explicitly asking for ratings. This manner of explicit asking could lead to explicit inference in the process of implicit learning, as would be in line with what Reber found. Therefore their results might be explained by participants using task strategies, and thus not reflect pure implicit learning. This means that we cannot be sure if the findings of Van Hamme and Wasserman's research are due to Error Driven Learning, or due to logic and reasoning.

The paper itself had some other problems as well. While they did find evidence *in the direction* of their hypothesis, only one out of thirty t-tests differed significantly from zero.

Added to this is that in working with cues such as food that predict an allergic or not outcome,

there will be certain pre conceived notions about certain foods. For example peanuts and walnuts are foods that are known for causing allergies and they were in the experiment. While they were not in the same food group (A, B, or X) they would give people a certain bias to an answer.

The goal of this paper is to find out which of these two mechanisms actually lies behind Error Driven Learning and whether or not it is an implicit process. Will we find learning in the absence of cues?

We will try to answer this question by replicating the findings that Van Hamme and Wasserman (1994) found with different cues and more participants. We want to reduce biases introduced into the experiment by using non-existing stimuli and increase the statistical power by increasing the number of stimuli and trial types. The latter will in turn help to mask the goal of the experiment and will thus help to reduce the amount of explicit reasoning taking place during the experiment as well.

We expect that by manipulating the presentation speed of the cues, we will find a difference in what participants learn in the slower presentation speed (room for explicit inference) and the fast presentation speed (forced implicit learning).

To model both the Rescorla-Wagner model and the Van Hamme and Wasserman model, we will use, amongst other papers, a paper by Hoppe et al. (n.d.). In this paper they go over the different ways to model Error Driven Learning with a simple neural network and on how to interpret the results.

We will use computational simulations with these models to find the amount of trials or stimuli that would work best for the experiment on human participants.

2 Computational Modelling

Before testing on human participants we first want to create models of the Rescorla-Wagner model and the Van Hamme-Wasserman model. Firstly we want to do this to see if the experiment they performed is actually capable of capturing the differences between the two models. The second reason to do so is to see the two proposed underlying methods of EDL at work and see on which aspects they differ and on which aspects they perform similarly.

To explain what exactly was modelled, we first explain the experiment of Van Hamme and Wasserman in detail. After that we go into the implementation and at the end we will see the results of these models on the experiment and compare them to the results of the original experiment as well.

2.1 The experiment

The participants in the experiment were told to imagine that they were an allergist that was trying to determine the cause of an allergic reaction shortly after their patient ate something. There were three different types of food a participant could see during a block. These were encoded as food A, B and X and were always shown as compound cues. X was seen in all of the trials, while A and B where both respectively seen in half the trials. Therefore participants saw either AX or BX in a trial. These food items were paired together with the outcome that indicated if an allergic reaction had taken place or not.

The participant then had to indicate how likely they thought any of the *three* foods could cause an allergic reaction. They had to indicate this every trial and there were 16 trials per block. The 16 trial always had the same order of cues; trial 1 was always AX, trial 2 always BX, trial 3 always BX and so on. Depending on the block the participant was in two things differed. The first one was the types of food that were filled in for A, B and X. So in one block they would see cheese, pork and blueberries, but in a different block they saw strawberries, peanuts and shrimp.

The other difference between blocks was the outcome conditions. There were three different outcome conditions, 0.00, 0.50 and 1.00, each of which the participants saw once. This condition reflects the probability of AX leading to the outcome (an allergic reaction) minus the probability of BX leading to the outcome. In the 0.00 condition both AX and BX have a 0.5 probability of leading to an allergic reaction. In the 0.50 condition AX has a probability of 0.75 and BX one of 0.25 and lastly in the 1.00 condition AX always predicts an allergic reaction and BX never predicts one. Therefore each participant saw 3 blocks of 16 trials each, with each block differing the type of food and the outcome condition.

2.2 Simulations

Modelling both the Rescorla-Wagner and the Van Hamme-Wasserman model was done in R (R Core Team, 2020), using the package `edl` (TBD) for implementation of the EDL formulas and the package `NDLvisualisations` (van Rij, 2018) for visualing the learning process trial by trial.

The main functions used from the `edl` package were `updateWeights` and `RWlearning`. These functions were originally designed for the Rescorla-Wagner equations and were adapted here for running the Van Hamme Wasserman model. The adjustments made to them and their original form can both be found in Appendix A.

To adapt the original functions a variable called `etaNeg` to `updateWeights`. `etaNeg` is false by default, which will result in the functions doing what they did before and preforming Rescorla-Wagner learning. However is `etaNeg` is set to true Van Hamme-Wasserman learning takes place. This means that now all the cues that are seen up until this moment are in the set of current cues, instead of only the cues that are seen at this moment. For each cue that is in this set of current cues, the same steps as Rescorla-Wagner learning are followed, as the change in weights between the cue and outcome when a cue is present does not change in the Van Hamme-Wasserman model. Each cue that is seen before, but is not in the current set of cues, the learning rate is multiplied by -1 (the eta is now negative, hence the name `etaNeg`). Then these cues are handled in the same way as the cues that are present.

The only thing that had to change in the `RWlearning` function was that it needed to be able to deal with `etaNeg` as well, as `RWlearning` calls `updateWeights`. This resulted in changing the name of the `RWlearning` function to `EDLearning` as the function did more than just implement Rescorla-Wagner learning now.

With these changes made to the code, the experiment could finally be modelled. The method of modelling the experiment for both the Rescorla-Wagner learning and the Van Hamme-Wasserman learning was the same, except for the value given to `etaNeg`. What we will describe here are the basics of the implementation, for the code itself see THIS LINK *.

A cue was created for each of the different food conditions (there were six in total) with an added background cue (for background cue see Rescorla, 1972). Cues for each of the outcome conditions were created as well. Each modelled participant runs 3 blocks of 16 trials. After each block the modelled participant has a saved "memory" of the strength of connection between cues and outcomes. Since the outcomes stay the same in all the blocks, we give this "memory" as a starting point for the next block.

Because it is not possible to model something that is *not* the outcome in the current version of the `edl` package, we modelled the outcomes as `Allergic` and `Not Allergic`. In the results we will look at these outcomes in two ways. One of them is the relative activation, which is when we subtract the connection strength of a cue to `Not Allergic` from the strength of that cue to `Allergic`, we will get the relative activation or activation difference. This is what would also be measured in participants responses in a forced choice task. The other way is just looking at the individual activation to `Allergic`, which is then a measure of the activation when there is not a two-way forced choice between `Allergic` and `Not Allergic`. The conceptual distinction between these two measures might not be immediately clear with this example. However, we will return to this point with different examples in Experiment 1.

*here is the link to the code so people can click and see for themselves

2.3 Simulation results

Figure 2.1 show the weights to `Allergic` minus the weights to `Not Allergic` for the Rescorla-Wagner model on the left and the Van Hamme-Wasserman model on the right. The division between the three blocks is indicated with a light grey, vertical dotted line and each of the foods that were shown in the experiment (plus the background cue) are seen in different coloured lines.

Although the strength of activations (i.e. model expectation) is different, qualitatively the two models make the same predictions.

We will take a more detailed look at the individual activation to `Allergic` instead of the relative activation we looked at just now. Figure 2.2 shows the weights to `Allergic` for the Rescorla-Wagner model on the left and the Van Hamme-Wasserman model on the right. Here too the division between the three blocks is indicated with a light grey, vertical line. The activations for the background cue (`env`) and the three foods that were seen in the first block are in colour, the rest of the foods is shown in grey.

What we see here is that in individual activation to `Allergic` there is a difference between the predictions the two models make. Namely in the Rescorla-Wagner model the activation of a cue that is not encountered again stays at the same weight, while the activation of a cue that is not encountered again in the Van Hamme-Wasserman model decreases in weight over time.

How do these models perform when comparing the results to the results of the original experiment? For the sake of readability - there are four original plots, each with then two comparison plots - these plots are shown in Appendix B. While the axis are different, in the original experiment they had to rate on a scale from one to eight, while here we have activation weights which are between one and zero, the overall patterns of the two models do not differ from that of the original paper. They also do not differ from each other, except on scale as we also saw in Figure 2.1

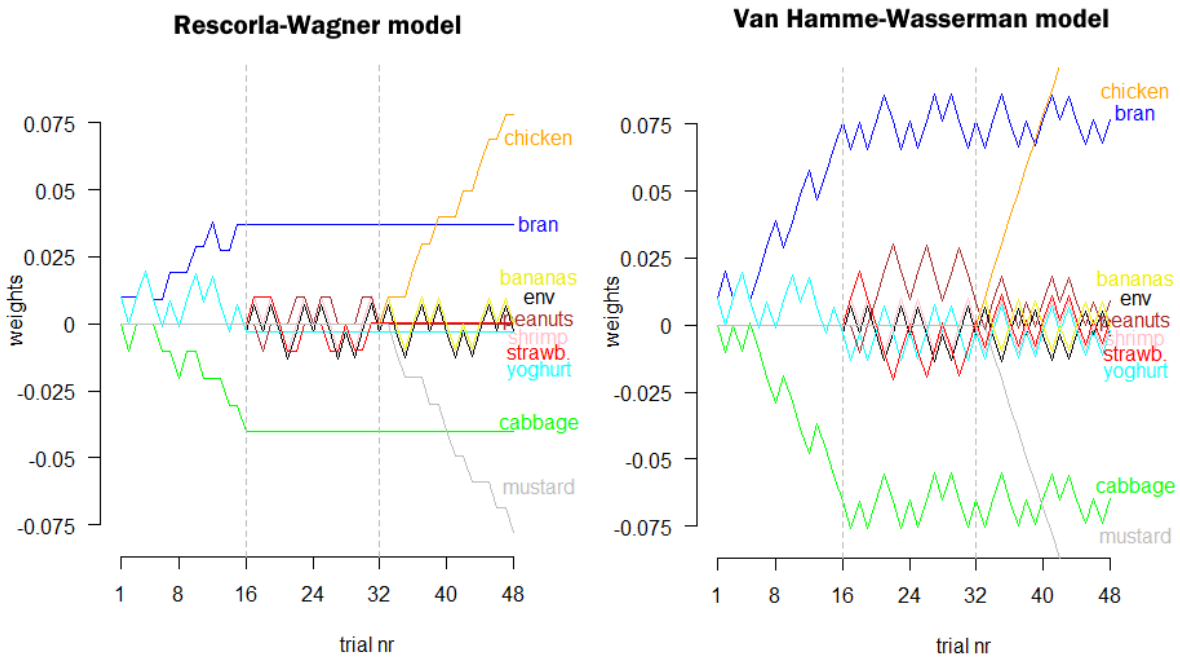

Figure 2.1: Weights to `Allergic` minus the weights to `Not Allergic` for each of the foods asked. Left: Rescorla-Wagner model. Right: Van Hamme-Wasserman model

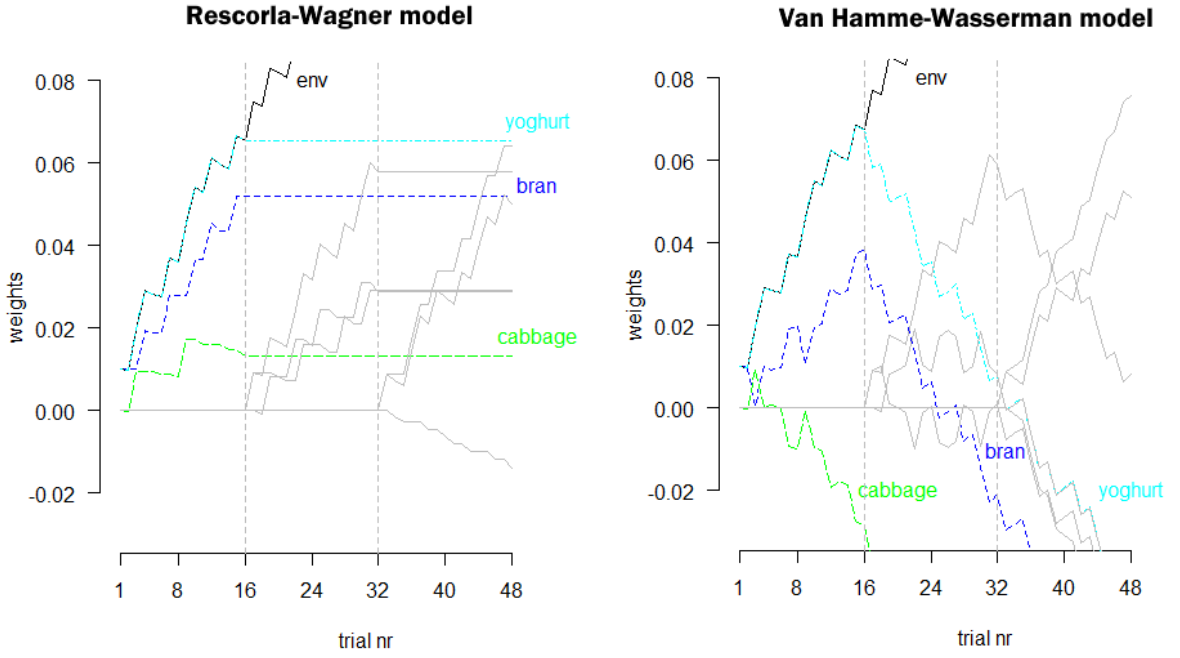

Figure 2.2: Weights to `Allergic`. Left: Rescorla-Wagner model. Right: Van Hamme-Wasserman model

2.3.1 Discussion

Van Hamme and Wasserman conceptualised the issue as learning in the absence of cues. However the simulation show a slightly more complex story, namely the models show no difference in relative activation (Figure 2.1 and Appendix B).

The only difference that emerges is the weight of the cues that are no longer encountered to `Allergic`, as could be seen in Figure 2.2. In the Rescorla-Wagner model these weights stay constant, but in the Van Hamme-Wasserman model they decline. A way to then test which of these two predictions is more likely is to introduce a test at the end of the experiment. One with both new and old stimuli, to compare if the activation of the old stimuli has decreased compared to when they were seen and to see if this is then lower than a novel stimuli that has not been seen before.

3 Experiment 1

The aim of this experiment is to test if the experiment performed by Van Hamme and Wasserman works .

We will do this by basing our experiment on the original experiment done by Van Hamme and Wasserman in their 1994 paper, while making adjustments based on the findings of modelling. This means including a test phase at the end of the three blocks (which are then the training phase), presenting different stimuli and reframing the context of the experiment.

3.1 Methods

3.1.1 Participants

3.1.2 Materials/Stimuli

3.1.3 Experimental Design

3.1.4 Procedure

3.2 Results

3.3 Discussion

Another adjustment that we will perform in this experiment is the changing of the stimuli. As already mentioned in the introduction, there might be preconceived notions about which foods will lead to an allergic reaction. We changed the cues to non-human objects and the outcome to finding a treasure or not.

- explain the experiment as a whole
- explain the exact stimuli, how they were chosen
- details such as implementation, length of seeing screens, the stimuli moving around a sort of circle
-

4 Experiment 2

4.1 Methods

4.2 Results

4.3 Discussion

5 Experiment 3

5.1 Methods

5.2 Results

5.3 Discussion

6 General Discussion

HERE COMES THE GENERALL OVERALL DISCUSSION

7 Conclusions

8 Acknowledgements

References

Ellis, N. C. (2006). Language acquisition as rational contingency learning. *Applied linguistics*, *27*(1), 1–24.

Hoppe, D. B., Hendriks, P., Ramscar, M., & van Rij, J. (n.d.). *An exploration of error-driven learning in simple two-layer networks.* (unpublished)

Hsu, A. S., Chater, N., & Vitányi, P. M. (2011). The probabilistic analysis of language acquisition: Theoretical, computational, and experimental analysis. *Cognition*, *120*(3), 380–390.

Kamin, L. J. (1967). Attention-like processes in classical conditioning.

Markman, A. B. (1989). Lms rules and the inverse base-rate effect: Comment on gluck and bower (1988).

R Core Team. (2020). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Ramscar, M., Dye, M., & Klein, J. (2013). Children value informativity over logic in word learning. *Psychological science*, *24*(6), 1017–1023.

Reber, A. S. (1989). Implicit learning and tacit knowledge. *Journal of experimental psychology: General*, *118*(3), 219.

Rescorla, R. (1972). Informational variables in pavlovian conditioning. In *Psychology of learning and motivation* (Vol. 6, pp. 1–46). Elsevier.

Rescorla, R., & Wagner, A. (1972). A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. *Classical conditioning ii*, *64*, 99.

Siegel, S., & Allan, L. G. (1996). The widespread influence of the rescorla-wagner model. *Psychonomic Bulletin & Review*, *3*(3), 314–321.

St. Clair, M. C., Monaghan, P., & Ramscar, M. (2009). Relationships between language structure and language learning: The suffixing preference and grammatical categorization. *Cognitive Science*, *33*(7), 1317–1329.

Tassoni, C. J. (1995). The least mean squares network with information coding: A model of cue learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *21*(1), 193.

van Rij, J. (2018). Ndlvisualization: Additional visualization functions for the ndl framework [Computer software manual]. (R package version 0.4)

Van Hamme, L. J., & Wasserman, E. A. (1994). Cue competition in causality judgments: The role of nonpresentation of compound stimulus elements. *Learning and motivation*, *25*(2), 127–151.

Widrow, B., & Hoff, M. E. (1960). *Adaptive switching circuits* (Tech. Rep.). Stanford Univ Ca Stanford Electronics Labs.

A Appendix A

Listing 1: original `updateWeights` function

```
function (cur.cues, cur.outcomes, wm = NULL, eta = 0.01, lambda = 1,
    alpha = 0.1, beta1 = 0.1, beta2 = 0.1) #hi
{
    bg <- getOption("background")
    cur.cues <- c(bg, cur.cues)
    if (is.null(wm)) {
        wm <- createWM(cues = cur.cues, outcomes = cur.outcomes)
    }
    else {
        wm <- checkWM(cues = cur.cues, outcomes = cur.outcomes,
            wm = wm)
    }
    Vtotal = 0
    if (length(cur.cues) <= 1) {
        Vtotal = wm[cur.cues, ]
    }
    else {
        if (ncol(wm) > 1) {
            Vtotal = colSums(wm[cur.cues, ], na.rm = TRUE)
        }
        else {
            Vtotal = sum(wm[cur.cues, ], na.rm = TRUE)
        }
    }
    Lambda = rep(0, ncol(wm))
    Lambda[which(colnames(wm) %in% cur.outcomes)] <- lambda
    lr = rep(eta, length(Lambda))
    if (is.null(eta)) {
        lr = alpha * (beta1 * Lambda + beta2 * (lambda - Lambda))
    }
    wm[cur.cues, ] = wm[cur.cues, ] + matrix(rep(lr * (Lambda -
        Vtotal), length(cur.cues)), nrow = length(cur.cues),
        byrow = TRUE)
    return(wm)
}
```

Listing 2: adjusted `updateWeights` function, everything below ### is new

```
function (cur.cues, cur.outcomes, wm = NULL, eta = 0.01, lambda = 1,
                              alpha = 0.1, beta1 = 0.1, beta2 = 0.1, etaNeg = FALSE)
{
  bg <- getOption("background")
  cur.cues <- c(bg, cur.cues)
  if (is.null(wm)) {
    wm <- createWM(cues = cur.cues, outcomes = cur.outcomes)
  }
  else {
    wm <- checkWM(cues = cur.cues, outcomes = cur.outcomes,
                  wm = wm)
  }
  Vtotal = 0
  if (length(cur.cues) <= 1) {
    Vtotal = wm[cur.cues, ]
  }
  else {
```

```
    if (ncol(wm) > 1) {
      Vtotal = colSums(wm[cur.cues, ], na.rm = TRUE)
    }
    else {
      Vtotal = sum(wm[cur.cues, ], na.rm = TRUE)
    }
  }
  Lambda = rep(0, ncol(wm))
  Lambda[which(colnames(wm) %in% cur.outcomes)] <- lambda
  lr = rep(eta, length(Lambda))
  ####################################################
  #New code

  m <- matrix() #create an empty matrix (matrix(rep) does not work in the VHW case)
  flag <- FALSE #flag since a completely empty matrix is not possible, so we remove the
  if(etaNeg == TRUE){
    for (i in 1:nrow(wm)){
      cues = rownames(wm) #for each row in wm, get the rowname and that is the cue
      if(cues[i] %in% cur.cues){ #if that cue is one of the current cues we can treat it
        if(all(dim(m) == c(1,1)) && flag == FALSE){ #is the first row?
          m <- matrix(lr * (Lambda - Vtotal), nrow = 1, ncol = length(Lambda), byrow = T
          flag <- TRUE
        }
        else{ #else rowbind it
          m <- rbind(m,matrix(lr * (Lambda - Vtotal), nrow = 1, ncol = length(Lambda), b
        }
      }
      else if(!(cues[i] %in% cur.cues)){ #if that cue is NOT one of the current cues we
        if(all(dim(m) == c(1,1)) && flag == FALSE){
          m <- matrix(lr * (-1) * (Lambda - Vtotal), nrow = 1, ncol = length(Lambda), by
          flag <- TRUE
        }
        else{
          m <- rbind(m,matrix(lr * (-1) * (Lambda - Vtotal), nrow = 1, ncol = length(Lam
        }
      }
    }
    matrix <- m
  }

  ####################################################################
  if (is.null(eta)) {
    lr = alpha * (beta1 * Lambda + beta2 * (lambda - Lambda))
  }

  ####################################################################
  if(etaNeg == TRUE){ #if you have VHW learning, we don't have to update only the curren
    wm = wm + matrix
  }
  if(etaNeg == FALSE){ #but if we have RW learning we want to only update the current or
    wm[cur.cues, ] = wm[cur.cues, ] + matrix(rep(lr * (Lambda - Vtotal), length(cur.cues
  }
  return(wm)
}
```

Listing 3: original `RWlearning` function

```
function (data, wm = NULL, eta = 0.01, lambda = 1, alpha = 0.1,
    beta1 = 0.1, beta2 = 0.1, progress = TRUE, ...)
```

```
{
    if (!all(c("Cues", "Outcomes") %in% names(data))) {
        stop("Specify a column Cues and a column Outcomes in data.")
    }
    out <- list()
    lout <- 0
    if (!is.null(wm)) {
        if (is.list(wm)) {
            out <- wm
            lout <- length(wm)
            wm <- wm[[length(wm)]]
        }
        if (!is.matrix(wm)) {
            stop(sprintf("Argument wm cannot be class %s: wm should specify weight matri
                class(wm)[1]))
        }
    }
    if (progress & (nrow(data) > 2)) {
        pb <- txtProgressBar(style = 3, min = 1, max = nrow(data))
        step <- min(c(max(c(1, round(0.01 * nrow(data)))), nrow(data)))
        for (i in 1:nrow(data)) {
            if ((i%%step == 0) | i == 1 | i == nrow(data)) {
                setTxtProgressBar(pb, i)
            }
            wm <- updateWeights(cur.cues = getValues(data[i,
                ]$Cues, ...), cur.outcomes = getValues(data[i,
                ]$Outcomes, ...), wm = wm, eta = eta, lambda = lambda,
                alpha = alpha, beta1 = beta1, beta2 = beta2)
            out[[length(out) + 1]] = wm
        }
        close(pb)
    }
    else {
        for (i in 1:nrow(data)) {
            wm <- updateWeights(cur.cues = getValues(data[i,
                ]$Cues, ...), cur.outcomes = getValues(data[i,
                ]$Outcomes, ...), wm = wm, eta = eta, lambda = lambda,
                alpha = alpha, beta1 = beta1, beta2 = beta2)
            out[[length(out) + 1]] = wm
        }
    }
    return(out)
}
```

Listing 4: adjusted `RWlearning` function, now called EDLearning

```
function (data, wm = NULL, eta = 0.01, etaNeg = FALSE, lambda = 1, alpha = 0.1,
                        beta1 = 0.1, beta2 = 0.1, progress = TRUE, ...)
{
  if (!all(c("Cues", "Outcomes") %in% names(data))) {
    stop("Specify a column Cues and a column Outcomes in data.")
  }
  out <- list()
  lout <- 0
  if (!is.null(wm)) {
    if (is.list(wm)) {
      out <- wm
      lout <- length(wm)
      wm <- wm[[length(wm)]]
```

```
    }
    if (!is.matrix(wm)) {
      stop(sprintf("Argument wm cannot be class %s: wm should specify weight matrix or l
                   class(wm)[1]))
    }
  }
  if (progress & (nrow(data) > 2)) {
    pb <- txtProgressBar(style = 3, min = 1, max = nrow(data))
    step <- min(c(max(c(1, round(0.01 * nrow(data)))), nrow(data)))
    for (i in 1:nrow(data)) {
      if ((i%%step == 0) | i == 1 | i == nrow(data)) {
        setTxtProgressBar(pb, i)
      }
      wm <- updateWeights2(cur.cues = getValues(data[i,
                                                     ]$Cues, ...), cur.outcomes = getVal

                           alpha = alpha, beta1 = beta1, beta2 = beta2, etaNeg = etaNeg)
      out[[length(out) + 1]] = wm
    }
    close(pb)
  }
  else {
    for (i in 1:nrow(data)) {
      wm <- updateWeights2(cur.cues = getValues(data[i,
                                                     ]$Cues, ...), cur.outcomes = getVal

                           alpha = alpha, beta1 = beta1, beta2 = beta2, etaNeg = etaNeg)
      out[[length(out) + 1]] = wm
    }
  }
  return(out)
}
```

B Appendix B

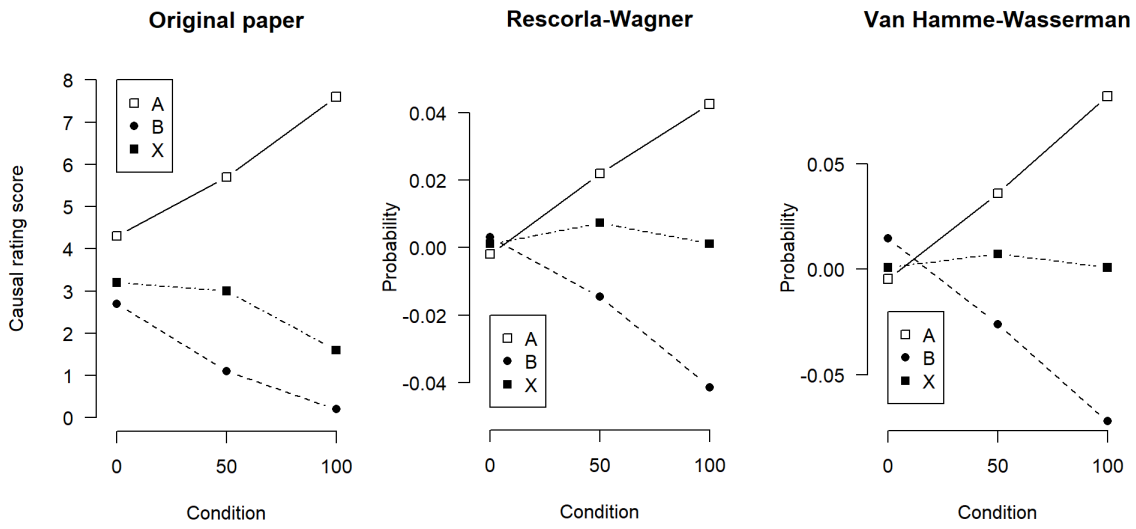

Figure B.1: Weights to `Allergic` minus `Not Allergic`. Middle: Rescorla-Wagner model. Right: Van Hamme-Wasserman model. Left: Results of the original paper.
Average for each condition

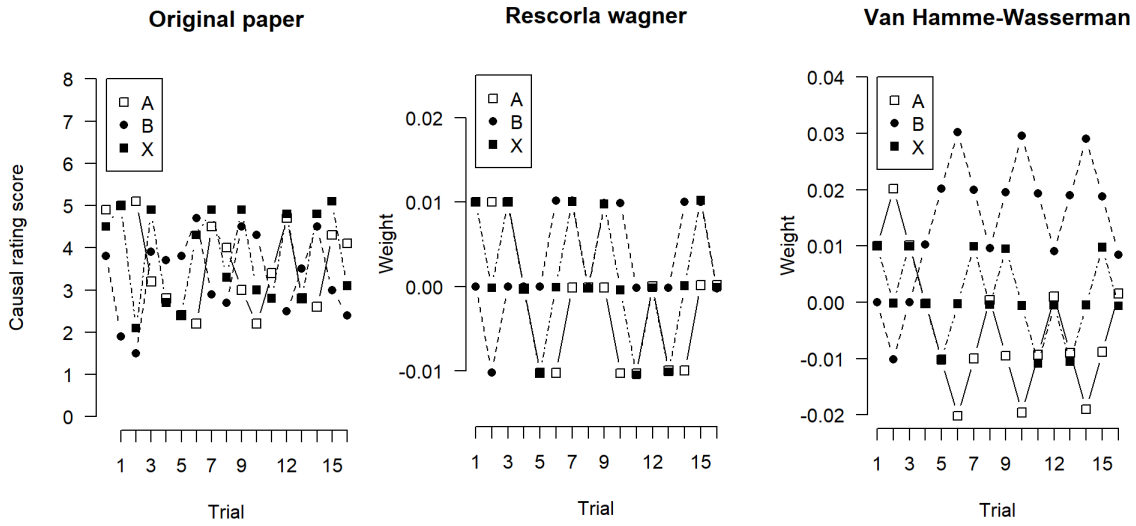

Figure B.2: Weights to `Allergic` minus `Not Allergic`. Middle: Rescorla-Wagner model. Right: Van Hamme-Wasserman model. Left: Results of the original paper.
Over trial for the 000 condition

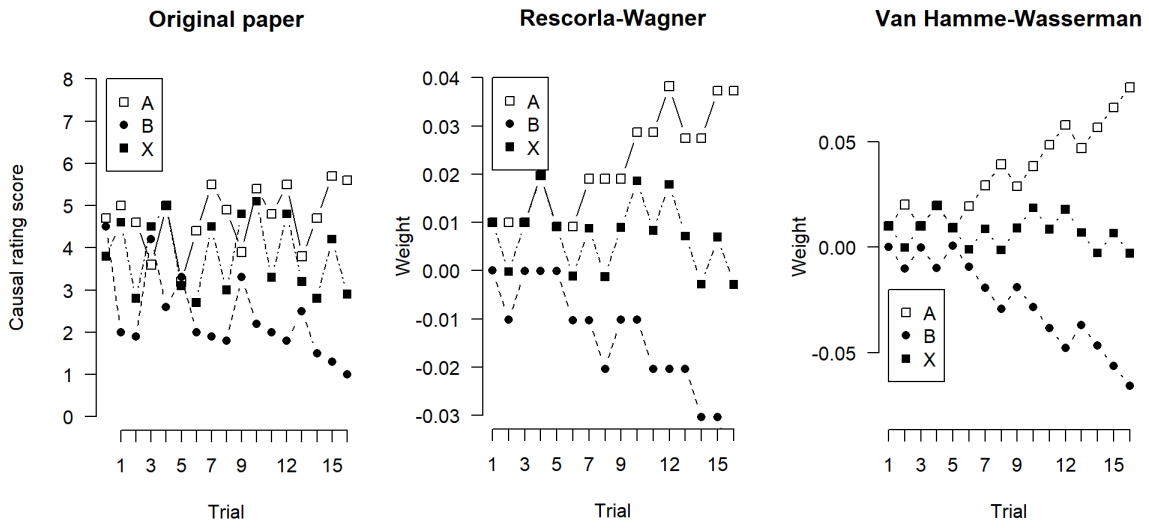

Figure B.3: Weights to `Allergic` minus `Not Allergic`. Middle: Rescorla-Wagner model. Right: Van Hamme-Wasserman model. Left: Results of the original paper.
Over trial for the 050 condition

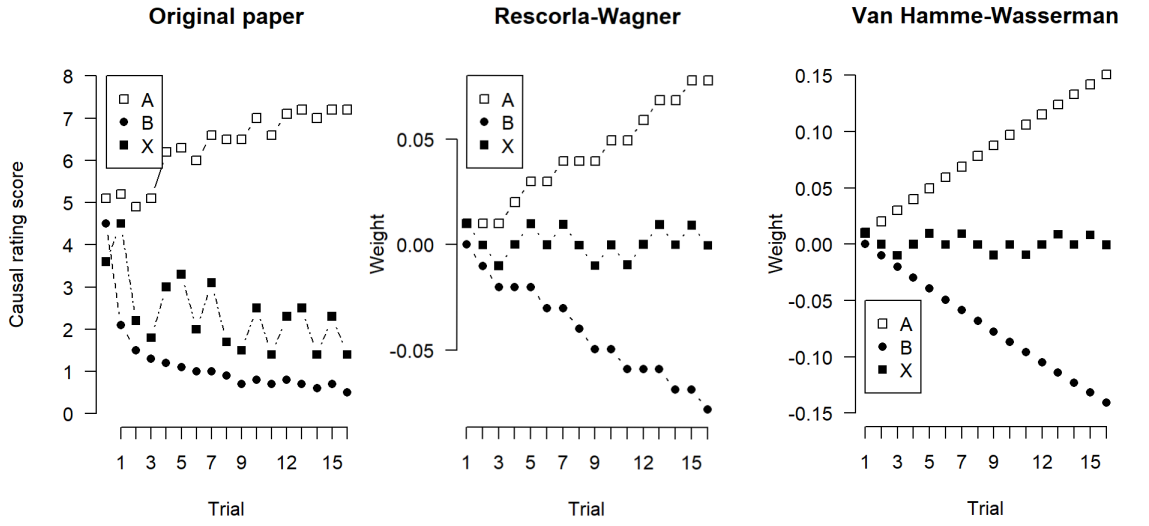

Figure B.4: Weights to `Allergic` minus `Not Allergic`. Middle: Rescorla-Wagner model. Right: Van Hamme-Wasserman model. Left: Results of the original paper.
Over trial for the 100 condition