university of groningen

faculty of mathematics and natural sciences

Analyzing Temporal Patterns in Behavioral Data with Liquid State Machines

Master's Thesis Behavioral and Cognitive Neuroscience

December 21, 2015

Student: Jurriaan Duyne (S1573683)

Primary supervisor: Dr. Marco Wiering

(*Department of Artificial Intelligence, University of Groningen*)

Secondary supervisors: Dr. Peter Lewinski, Marten den Uyl

(*Vicar Vision*)

Contents

1 Introduction **1**
- 1.1 Background . . . 2
  - 1.1.1 Behavioral Marketing . . . 2
  - 1.1.2 Machine Learning . . . 3
- 1.2 Aims and Research Questions . . . 6
- 1.3 Outline of the Thesis . . . 8

2 Methods **9**
- 2.1 FaceReader . . . 9
- 2.2 Data . . . 11
- 2.3 Behavioral Marketing Analysis . . . 15
- 2.4 Predicting Behavioral Intentions with Machine Learning . . . 16
- 2.5 Supervised Learning Algorithms . . . 20
  - 2.5.1 Pseudoinverse . . . 20
  - 2.5.2 Artificial Neural Networks . . . 22
  - 2.5.3 Support Vector Regression . . . 25
- 2.6 Feature Selection . . . 28
- 2.7 Summary . . . 29

3 Liquid State Machine **31**
- 3.1 Recurrent Neural Networks . . . 31
- 3.2 Reservoir Computing . . . 33
  - 3.2.1 Echo State Machine . . . 35
- 3.3 Liquid State Machine . . . 37
- 3.4 Generation and Activation of a Liquid . . . 40
- 3.5 Liquid Architectures . . . 42
  - 3.5.1 Waxman . . . 43
  - 3.5.2 Scale-Free . . . 44
  - 3.5.3 Watts-Strogatz . . . 46
- 3.6 The Edge of Chaos and Learning in the Liquid . . . 47
- 3.7 Inspecting Temporal Patterns . . . 50
- 3.8 Summary . . . 51

4 Results **53**
4.1 Classical Analysis 53
4.2 Machine Learning Regression and Setting a Baseline 56
4.3 LSM 58
4.4 Feature Selection 63

5 Conclusion **66**
5.1 Suggestions for Future Research 67

Bibliography **69**

Abstract

We propose the use of brain inspired machine learning to complement statistical tests currently used in the field of behavioral marketing research. We analyze a data set of facial emotion responses from participants who watched a commercial. Our goal is to predict behavioral intentions of the participants from the *temporal patterns* of their facial emotions. We suggest interpreting the data as a regression problem with a temporal component, allowing machine learning algorithms to discover *which* facial emotions are important for behavioral intentions and *when* these features play a decisive role. We first set a baseline performance collapsing the temporal component of the regression analysis by taking the average facial emotions over the whole video or by taking the difference between the averages of the first and second half of the commercial. Thereafter, we apply a Liquid State Machine (LSM) to analyze temporal patterns. The LSM contains a recurrent neural network (RNN), which produces states with implicit temporal information and higher order combinations of features. The states of the RNN are regressed onto behavioral intentions with Support Vector Regression (SVR). We apply forward and backward feature selection to explain which facial emotions have most impact on predicting the behavioral intentions. Moreover, at several points in time we inspect the performance of the LSM to expose when important effects during the commercial take place. Optimizations in the notoriously difficult parameter search for the LSM are suggested. We investigate several different topologies to improve the consistency of the general performance of the RNN. Results suggest that "Happiness" is the most important facial emotion for the investigated dataset and that the optimal decision time to predict behavioral intentions is at 80% in the commercial. We compare our results to traditional statistics in behavioral marketing and show the added value of using machine learning to include analysis of temporal patterns. Finally, we suggest that our proposed methods can be applied in similar fields in which data with complex temporal patterns are found.

1 Introduction

In neuromarketing and behavioral marketing research, experiments are conducted in which rich temporal data are recorded from participants. Researchers investigate participants' reactions to commercials, such as facial expressions (changing over the duration of the commercial) as well as self-assessed behavioral responses (static post hoc questionnaire answers in which participants express their attitudes and buying intentions). However, when statistical analysis is applied, data are simplified in order to test hypotheses. For instance, it could be proposed that if people are more happy during a commercial, they are more likely to buy a product seen in that commercial. To test this hypothesis the average level of happiness over the whole duration is presented, ignoring any potential temporal patterns that could be important for viewers' behavioral intention. Although this simplification leads to insightful conclusions, potentially significant details are lost in the process.

In this thesis, we propose a machine learning (ML) approach to extend and improve the analysis of behavioral time-series data. We attempt to discover complex temporal patterns in facial emotions which we can use to predict participants' behavioral intentions. Before extracting temporal clues, we try to predict the behavioral intentions from the average emotions over time and from some other transformations which simplify the temporal aspect. After setting this baseline, we utilize the Liquid State Machine (LSM), which extracts meaningful temporal information. We describe a specific dataset in detail as an example and first apply the standard statistical data analysis. We compare this standard statistical analysis with the ML approach and see how the latter can be informative to a behavioral marketing researcher. Finally, we suggest optimizations to the ML paradigm in order to efficiently employ the methods under investigation.

This chapter will provide the background setting and develop the justification for the aforementioned introduction of ML into behavioral marketing research. We describe our aims, leading up to exact research questions. Finally, an outline for the remainder of the thesis is given.

1.1 Background

1.1.1 Behavioral Marketing

The field of behavioral marketing research is chiefly concerned with delivering "persuasive communication convincing another party to change their opinion or attitude" (Meyers-Levy and Malaviya, 1999). In order to arrive at the optimal means of persuasion it is desirable that we understand consumers' behavior in relation to an advertisement. This can be achieved by analyzing self-report responses, but it would be advantageous to grasp or even predict physiological responses that lead to particular behavior. This has led to the development of the field collectively referred to as neuromarketing, in which techniques are applied such as brain imaging (Lewis and Phil, 2004; McClure et al., 2004), EEG (Vecchiato et al., 2011), eye-tracking (Khushaba et al., 2013) and heart-rate monitoring (see Lewinski et al., 2014b, for a more extensive overview). By relating physiological responses to behavioral intentions, researchers make important discoveries about the subconscious aspect of decision making.

All of the mentioned studies contain a temporal aspect in their recordings, which may be lost depending on the analysis. We will examine a concrete example of the type of data in which information from temporal data is lost. In Lewinski et al. (2014b), the authors analyzed reactions of participants to a commercial advertisement. They predicted the effectiveness of the advertisement by analyzing Facial Expressions of Basic Emotions (FEBE, see Ekman, 1972) with FaceReader (Noldus, 2013). In Chapter 2 we will elaborate on these methods and data. The facial expressions were analyzed every frame of the video and thus entail rich information about temporal changes of emotions. Specifically, the 6 basic emotions were examined per participant per frame of video recording. Unfortunately, these data are too complex to perform straightforward statistical tests on, so the tests need to be applied

on averages over the temporal sequence. After constructing three different conditions, low, medium and high amusement, the authors found that there were significant differences in the "global mean average score of top 10% peak values of facial expressions of happiness" within these conditions. They also found a positive correlation between the facial expression happiness and the self-report attitude towards brand and attitude towards the ad, indicating that the detected happiness was also reflected in the participants' attitude.

Similar to the latter study, facial expressions and attention span were investigated in order to analyze how long and how entertaining commercials should be (Teixeira et al., 2012, 2014). Though the hierarchical bayesian model which was applied in these studies is more sophisticated, again there is no place for the temporal pattern of emotion other than the absolute duration and the average happiness of the participant. The mentioned findings are certainly insightful, but two aspects considering these data can be improved. First, temporal information was lost in the examples because averages were used to compute the differences between conditions and the correlations. Second, results were found because they were hypothesized with theoretical considerations and led by a particular research bias. A research bias in itself is not problematic, because the researchers' expectations were well founded in theory. However, if we wish to alleviate the restriction of applying statistical tests which make sense because the theory predicts they will make sense, we should traverse another route.

1.1.2 Machine Learning

We argue that in order to extract insightful data from the temporal aspect of time-series data, machine learning should be introduced. With machine learning we can discover data-driven effects instead of effects predicted from theory. To be able to do this, we need to take a different perspective on how data should be analyzed: instead of applying statistical tests on averages of emotions over time, we should develop a model by learning patterns from a training set, a large part of a data set, after which we can predict the labeling of a test set, the small remainder of that data set. Specifically, we would like to train an algorithm to abstract generalities from the data and use these to predict the class or response of an unseen

example. In theory this can be accomplished applying regression analysis from statistics, for example a general linear model, on the training set and then extrapolate to the test set. However, with machine learning algorithms we can create more sophisticated (arbitrarily complex) models (e.g., neural networks or support vector regression) and even those that can keep into account the temporal dimension (the LSM). In addition, these algorithms do not assume some of the mathematical properties associated with statistical regression. For obtaining baseline results from regression[1] we first model the emotions averaged over time in a machine learning paradigm: the averaged responses in a training set of participants are regressed onto their behavioral intentions and then this regression model is used to predict the behavioral intentions of a test set.

Perhaps the best-known machine learning technique for classification and regression is the artificial neural network (ANN). ANNs are information processing paradigms that are inspired by the nervous system. One of the first ANN algorithms for classification was the perceptron (Rosenblatt, 1958), which is a linear classifier that forms the basis for the multilayer perceptron and other ANNs with more complex architectures that can classify nonlinear problems. However, due to the lack of a proper training algorithm, multilayer perceptrons did not gain popularity until it was discovered that back-propagation could be used for training these networks on data (Werbos, 1974; Williams et al., 1986). Even with back-propagation, the basic multilayer perceptron popularity stalled because it turned out classification boundary optimization by Support Vector Machine (SVM, see: Vapnik, 1995; Cortes and Vapnik, 1995) outperformed the ANN. Similarly, for regression Support Vector Regression (SVR) (Vapnik et al., 1997) was introduced.

In recent years, with the advance of deep learning, in which many hidden layers are efficiently trained to create highly abstracted representations, the neural network has returned as a major technique in image classification (Krizhevsky et al., 2012) and speech recognition (Hinton et al., 2012). The hidden layers are augmented with convolutions and max pooling to make them more computationally efficient and add invariance to the location of particular

[1]Throughout the text, regression is not used in the strict statistical sense. Regression simply refers to having a model to predict results of the dependent variable

features in an image. These functions were inspired by the visual areas in the brain and turned out to be very effective for object recognition. A difficulty of deep neural networks in the context of behavioral data is that they are not directly usable on temporal sequences unless they are combined with a recurrent neural network. Mostly, the long short-term memory (Hochreiter and Schmidhuber, 1997) is used in the literature (e.g. Sutskever et al., 2014), adding an additional layer to the already extensive deep net. Moreover, a major issue can arise due to the complexity and therefore the increasing risk of overfitting: these models have to be trained on many data in order to have sufficient generalizability. Lastly, training the deep nets is relatively costly compared to techniques such as the SVR, although this has become less of a problem due to recent increases in computational speeds and optimizations in training (Hinton et al., 2006; Martens, 2010).

It is important to note the classification or regression algorithms are neutral toward the temporal aspect of the data. The models can be based on averages over a temporal sequence or on values of other transformations that lose temporal detail. In order to address the issue of time dependent effects we opt for a liquid state machine (LSM; Maass et al., 2002). The LSM contains a recurrent neural network (RNN) which is well suited for temporal input data. It outputs a temporal sequence but this sequence includes implicit information from past inputs, making it suitable for regression based on multiple previous time steps. Moreover, it allows us to track the accuracy of regression throughout the temporal sequence of the data presented, similar to how multiple votes can be used to increase the accuracy of the prediction from a complete time series. Similar to the Echo State Network (see Jaeger, 2001), which was introduced around the same time, the LSM's internal RNN, or liquid, is never explicitly trained. The liquid's activation turns out to be complex enough to provide a higher order mapping in which an SVM or SVR can learn which activation represents which class or value. In other words, in the LSM construction, the RNN transforms the data (it is a non-linear mapping of the features with an implicit time component) and is then read out by SVR at several pre-determined points in time.

Unfortunately, there is a drawback of using machine learning in the analysis of this type of psychology experiments: the number of participants is relatively low, so although there is

a multitude of data per participant, one participant is only one example of the particular condition or behavioral intention. It is considered to be better to have many examples (i.e. a large training set) because then the algorithm will be better at discovering generalities instead of storing specific differences between the viewed examples (overfitting, mentioned as early as Rosenblatt, 1958). Moreover, for the LSM the lack of training in the liquid does pose a downside: for the same initialization parameters the behavior of the liquid on identical data may be very different. Sometimes a liquid can provide very good separation of the input and hence good performance for classification or regression, although there is no guarantee that it will do so every time after random initialization. There are ways to combat this variation in performance and we will review some possibilities in Chapter 3.

Nevertheless, the ML algorithms will be useful because the algorithm's level of performance is directly affected by how well they can detect and learn from correlations between the data set and the target variable. Without any correlation between facial features and behavioral intentions, it would not be possible to make a better prediction than the mean of the intentions when inspecting a participant. Conversely, if there is a correlation it should be possible to learn from it, which allows us to make specific predictions. For instance, if one feature is more important than another, the performance of the classification or regression algorithm should be impacted more if we omit the former than if we omit the latter. This procedure is called feature selection and can be wrapped around classification or regression algorithms (Kohavi and John, 1997). Basically, the whole algorithm is ran multiple times with different features left out/in to compare the performance of those features. Using feature selection, we can work backward to understand the correlations in the data after we know which features in the data are important for achieving good performance in the ML paradigm.

1.2 Aims and Research Questions

In the previous sections we have discussed the current use of data in behavioral marketing and have informed the reader of some possibilities using machine learning algorithms. The aim of this thesis is to apply these machine learning capabilities to the field of behavioral

marketing, specifically to make sense of temporal data. Feature selection can tell us *which* features are important and tracking the accuracy of the regression through time can inform us about *when* the regression results are optimal. We will discuss the use and performance of the LSM and explain how it can complement and enhance statistical hypothesis testing for time-series analysis in behavioral marketing. We do not aim to instigate a paradigm shift; we suggest, however, that the proposed techniques can point researchers into a particular direction, discovering correlations that previously had been untouched (e.g. a certain sequence through time of a combination of features). In other words, machine learning allows for a data-driven approach to facial features inspection.

The previous aims can be summarized in the following research questions:

1. Can we use a Liquid State Machine to interpret behavioral data? This follows directly from the purpose of this thesis. More generally, we will try to demonstrate to which extent it is useful to apply machine learning in a behavioral research setting.

2. Does the analysis of behavioral data with an LSM benefit from feature selection and from tracking the LSM's accuracy through time? Specifically, it should be estimated which are the most accurate slices in the temporal sequence, similar to temporal voting. Moreover, we inspect if the best features are most relevant in that setting.

3. Can we minimize the parameter search for finding a well performing liquid by choosing a particular topology or applying a learning rule to update the connections in the liquid? The main practical obstacle with LSMs is that for the same initialization parameters the RNN may perform differently. We see if this problem can be reduced by choosing particular robustly performing topologies or if learning in the liquid can produce stable performance over several different initial parameters. This would significantly reduce the parameter search time.

A researcher might discover unpredicted correlations between complex variables by applying smart transformations and comparing many variables. The use of machine learning not only facilitates this process, it allows this search where human capabilities seize to make sense of the data. At the same time, this could be viewed as a downside because some

discovered correlations may not adhere to the researcher's intuition or fit into a particular theoretical framework. Finally, if the results from the machine learning approach agree with theoretical expectations, it will be a strong confirmation of the theory, as no other coincidental correlations in the data have caused the hypothesized effect.

1.3 Outline of the Thesis

After discussing the background and aims in this chapter, we will discuss facial emotions and preprocessing from a particular data set in detail, including details of the classical (statistical) analysis of the data in Chapter 2. Moreover, in that chapter we discuss how averages over a temporal sequence can be used for machine learning and how transformed features can emulate researchers searching for correlations. Detailed descriptions of the ANN and SVR algorithms are given. The liquid state machine and its variations will be discussed in depth in Chapter 3; it is explained how its performance is expected to be influenced by particular optimizations and how it can be used to interpret behavioral data. Finally the results with the classical techniques and the results of applying the LSM to the behavioral data and applying feature selection and temporal inspection will be presented in Chapter 4 and discussed in relation to the classical analysis in Chapter 5.

2 Methods

As this thesis is meant to introduce tools for analysis in the behavioral sciences, it will be important to discuss the methods in technical detail. We begin by describing how data are acquired for our leading example data set with the automated facial expression recognizer FaceReader. Then, we will briefly inspect the techniques mostly used in the behavioral marketing literature. We will start by describing supervised learning in general. To this end, we will introduce the machine learning paradigm and discuss the pseudo inverse, artificial neural networks and support vector regression. With these techniques we can set a baseline performance level to which we can gauge the performance of the liquid state machine: first we introduce the supervised learning paradigm with a simplified time dimension before we extend the analysis to include complex temporal pattern extraction. The time dimension is simplified in several ways, from omitting any facial feature to simulating researchers' intuitions (hard-coding transformations that are expected to be important). After supervised learning is introduced, the algorithm for feature selection will be given. The aim of this chapter is to write an understandable synthesis for the behavioral marketing and machine learning fields.

2.1 FaceReader

The data for this project come from an experiment from a psychological neuromarketing study, in which the researchers analyzed emotions from participants' facial expressions. The automated facial emotion analysis was performed by FaceReader (Noldus, 2013). At each frame, FaceReader fits an active appearance model (Cootes et al., 2004) onto a face, which gives a feature vector of landmark points on a face relevant for expressions. This approach is called holistic because it does not model local characteristics of a face in detail but instead

codes a representation in which the relative positions of landmarks are given. Although it seems desirable to encode small variations relevant to emotions, the global model is more generally applicable because it relies less on high resolution content and is more robust to changes in lightening or pose (Den Uyl and Van Kuilenburg, 2005).

In order to automatically analyze emotions, it is assumed there is a relation between facial expressions and emotions as suggested in (Ekman and Keltner, 1970). Although researchers acknowledge that not all individual expressions relate to one single emotion at the time and that there are great individual differences, a starting place had to be chosen. In FaceReader, emotion classification was trained in a supervised manner: an artificial neural network was trained on a database of input pictures labeled for an emotion. Good performance was achieved for the Karolinska database (Lundqvist et al., 1998) and it was later revalidated on new data, which indicated that the emotion recognition performance of FaceReader was comparable to that of human performance: 88% correct compared to 85% (Lewinski et al., 2014a). For more technical details on the active appearance model fitting and neural network architecture see, Den Uyl and Van Kuilenburg (2005); Van Kuilenburg et al. (2005).

Output from FaceReader consists of many features: six basic emotions (Ekman and

Figure 2.1: The six basic emotions according to Ekman and Keltner (1970): Anger, Fear, Disgust, Surprise, Happy and Sad

Keltner, 1970, also see Figure 2.1) are augmented with a neutral facial expression, and since later versions of the software, contempt, valence (measure of the attitude of the participant) and arousal (measure of the activity of the participant) (VicarVision, 2015). The outputs consist of numerical values between 0 and 1 for each emotion, except for valence which ranges from -1 to 1. In addition, 20 Action Units from the Facial Action Coding System (FACS) (Hager et al., 2002) are classified from the active appearance model. This amounts to 30 features per frame of analysis. Unfortunately, for this project the FACS action units were not suitable, as their activations were too sparse to meaningfully connect to a regression algorithm. As a result the features used are the six basic emotions plus neutral, contempt, valence and arousal. According to the publisher ,"FaceReader is used worldwide at more than 300 universities, research institutes, and companies in many markets, such as consumer behavior research, usability studies, psychology, educational research, and market research" (Noldus, 2015). A screenshot from the product is shown in figure 2.2.

2.2 Data

As mentioned earlier, the data to which we apply machine learning come from a neuromarketing study (Lewinski et al., 2016). A group of 150 participants viewed a 30-second Doritos commercial during which their faces were recorded with FaceReader. The study involved so-called avatars: artificial depictions of a face that could display facial expressions during the video. There were two conditions: a control condition without an avatar and an interfering condition in which the avatar had incongruent expressions with regard to the video. Concretely, this means that during an amusing part of the video, where people would normally smile or laugh, the avatar looks like it is disgusted. Figure 2.3 shows an example of the two conditions.

Prior to watching the video participants were asked questions to assess their affective and cognitive state and a priori attitudes toward what was shown. Afterward, their responses to the video material were inquired, which are called behavioral intentions (BI). For the purposes of this project, we investigate their attitude toward the ad (AAD), attitude toward the brand (AB) and purchasing intentions (PI), which were collected by following the Advertising

Figure 2.2: Screenshot from FaceReader 6 (VicarVision, 2015) during the analysis of a commercial. The video at the top left was watched by the participant in the top right of the screen. The face model can be seen as a mesh overlaying her face. At the bottom of the screen the changes of emotions throughout the video are displayed. Notice that these are averages for all participants of this group project. The high value for happiness can certainly be seen at the participant under current consideration

Effectiveness Model (Olson and Mitchell, 2000), see Lewinski et al. (2014b) for more details. The participants judged the questions on a 7-point Likert scale where 1 indicated least favorable attitudes or lowest purchasing intentions and 7 indicated most favorable attitudes or highest purchasing intentions. We do not discuss further details about the behavior marketing models because that is not the aim of this thesis; the point is that we need to know the format of the data because we will try to predict the behavioral intentions from the facial emotion expressions.

(a) Control condition

(b) Incongruent avatar

Figure 2.3: Screenshot from the Avatar Experiment in Lewinski et al. (2016). In the control condition they only present the video, whereas an incongruent avatar was added in the experimental condition.

Unfortunately, not all data were usable and we have made a selection of participants. First we established some participants did not have facial responses for the full duration of the ad or that their behavioral data were missing. As a minimum, it was required that there were 100 frames available to analyze, leaving 135 participants. FaceReader most likely had difficulty mapping the face model due to lighting conditions or unexpected pose changes. In addition, there was no prior emotion bias recorded per participant which could be used to normalize their neutral expression. For some participants this led to the data being dominated by one particular emotion, for instance anger or surprise. In some cases, the output of these emotions was larger than 0.95 during more than 90% of the video; these cases were removed as this was very likely due to an error in FaceReader. As there was no way of estimating whether the participants had a neutral face in the first part of the video (likely they had not due to the surprised reaction to the avatar) it was impossible to calculate our own bias to subtract from the later data. Finally, if participants had more than five consecutive seconds missing (failed fit in FaceReader) out of the 30 seconds of data recording, they were also removed. Thus, another 14 participants were removed leaving 121 participants: 62 in the control condition and 59 in the hindering condition.

Finally the facial emotion data were normalized per feature over all remaining participants so that the data would be proportionally scaled between 0 and 1 over the entire dataset. This was achieved by applying the following equation:

$$X_{norm} = \frac{X_{init} - min(X_{init})}{max(X_{init}) - min(X_{init})} \tag{2.1}$$

where X_{init} is the initial vector of all the values of one particular feature for all time steps for all the participants and X_{norm} is the normalized one.

2.3 Behavioral Marketing Analysis

Previous studies have demonstrated that facial expressions can be used to predict behavioral intentions and attitudes (Lewinski et al., 2014b). Moreover, facial mimicry is thought to modulate a person's own facial expressions. Thus, it is hypothesized that displaying avatars with particular facial expressions will influence the attitudes and intentions of consumers watching them at the same time as they view a commercial. Hence, we can think of the incongruent condition as interfering with consumers' own responses: they are expected to show fewer facial expressions of happiness in the hindering condition compared to when the control video is displayed. Consequently, the attitudes and intentions should likewise be diminished when viewing the interfering expressions. So we can set up a hypothesis:

H_1: There is a significant decrease in facial happiness expression in the hindering condition compared to the control condition.

Initial analysis shows that the average happiness of the participants is not distributed normally, so this hypothesis could not be tested by straightforwardly running a t-test to see if the difference between the groups was significant. Instead, the Mann-Whitney U test will be performed. Even if there is no significant difference between these conditions there might very well be a relationship between the expressed happiness and the Behavioral Intentions. We follow Lewinski et al. (2014b) in positing that the facial expression happiness will correlate positively with behavioral intentions:

H_2: An increase in facial happiness will correlate with increased attitude toward the ad, attitude toward the brand and purchasing intention.

In testing this hypothesis, we calculate Spearman's rank order correlations between the facial expressions of happiness and behavioral intentions. In addition, we briefly investigate if there are effects of other facial emotions on behavioral intentions. We choose the most

likely ones to show some effect: Happy, Neutral, Surprised and Valence. Finally, we inspect if differences between the first and second half of the video in terms of these four emotions correlate with the behavioral intentions. Thus, we are applying a "smart" transformation to the data in order to extract information from the temporal aspect, though within certain limits. This can be seen as a theory-driven hypothesis, realized by applying transformations on and averaging emotions over the temporal sequence. In the next section we will explain how we can simulate the theory-driven expectations as input for a machine learning algorithm. In the discussion section, we compare and contrast theory-based results to data-driven results which were found by employing the LSM.

2.4 Predicting Behavioral Intentions with Machine Learning

As we detailed in the introduction, we aim to introduce a machine learning paradigm to analyze behavioral data. This entails making the analysis suitable for extracting information about which features are important at which point in time. In order to do this, we opt to turn to the LSM, as explained in the next chapter. However, before moving to the intricacies of the LSM we first need to discuss how a paradigm for predicting the dependent variable can be a helpful and meaningful means of analysis for behavioral data.

The aim is to make a regression model from the facial emotion features of a subset (training set) of participants to their behavioral intentions. To predict behavioral intentions, the simplest estimate we can conceive of is to use the average behavioral intentions over all participants; the error is simply the mean squared error. However, we can train models to learn the relationship between facial expressions and behavioral intentions. The techniques we employ are called supervised learning techniques because we know the outcome when we train the regression model. This model will serve to predict the behavioral response of the remainder of the data (test set), which can then be compared to the actual behavioral intentions of the participants in the test set. The performance on the test set gives an indication on how well the reaction of new (unseen) participants can be related to their behavioral intentions. If the error is really low, we can be confident that we have a good

model to predict the behavioral intentions after inspecting the facial emotions; if the error is close to the error that results from using the mean as a predictor, it is clear that the model is not more informative than just estimating the mean response. In the section on supervised learning algorithms, we will detail how the regressions models are constructed with machine learning algorithms. We will briefly start from linear regression with a pseudo inverse, and then move on to introduce the machine learning techniques ANN and SVR.

But which representation of the facial features must we use to apply the regression on? Most straightforwardly, predictive regression models can be trained on the averages of the facial emotions over the whole temporal sequence, just as the means and correlation in the statistical analysis from the previous section. To predict a behavioral intention, the average happiness, sadness, etc., are put into a trained model and a value for the behavioral intention of the participant under consideration is calculated. In addition, the models can also be made from other transformations such as the ones discussed in the statistical analysis section. For the regression task, we do not keep into account possible effects of the experimental condition –which separates two groups of participants into different conditions– although it was hypothesized that it would influence the facial emotion expressions and thus influence the relationship between facial emotion expressions and behavioral intentions. Nevertheless, we can expect that if the algorithm has a high enough complexity, it may learn to implicitly distinguish between conditions and give proper predictions for the behavioral intentions.

The data set is divided into an 83%-17% split for training and testing respectively: 101 participants are used for training and 20 for testing. To set the internal weights of the ANN, the training set is split again to create a validation set resulting in what effectively is an 66%-17%-17% split of training, validation and testing. Moreover, sixfold cross-validation is used for both ANN and SVR, where each time another 20 participants were selected randomly, but without replacement, so that each part of the data was at one point used for training and at another point for testing. For SVR, to set the parameters fivefold cross-validation on the training set is used, leading to embedded cross-validation.

As the error measure the Mean Squared Error (MSE) is used:

$$\frac{1}{n}\sum_{i=1}^{n}(r^i - y^i)^2 \tag{2.2}$$

The MSE is used for learning by the ANN, and explicitly for testing on the ANN and SVR test sets. The learning algorithms produce y^i, predicted values for the test set, which are compared to r^i, the real (target) values of the behavioral intentions. The results we report are the MSE calculated over the test set. To asses the stability and to test if the differences are significant, the embedded cross-validation procedure ran 50 times, each time selecting different random data splits, albeit the same split for each algorithm in the same iteration. Paired t-tests with Bonferroni correction should indicate whether there are significant differences between the following baseline data manipulations.

To compare results from our ML approach directly to the statistical analysis, we will first see how well we can predict the behavioral intentions by using three theory-driven feature sets that collapse the temporal dimension to mimic the statistical analysis:

1. The mean behavioral intentions over all participants in the training set;
2. Averages of the facial expressions calculated over the whole temporal sequence for each participant;
3. Theory based selection and transformations of averaged facial emotion features which add extra (but still implicit) information about the temporal aspects of the data set.

The first item is the best that an ML algorithm could learn if there were no relation between the facial emotions and behavioral intentions. It could simply learn the average behavioral intention value of participants in the training set. Therefore, as first predictor we use the mean, without any training of the facial emotions. We calculate the Mean Squared Error as follows:

$$\frac{1}{n}\sum_{i=1}^{n}(M_{train} - r^i)^2 \tag{2.3}$$

M_{train} is the mean of the training set and r^i are the target values. As we know that there are correlations we expect to see a significant improvement by learning to predict the behavioral intentions from the second item, averages of facial emotions. The second item is to have a direct comparison to the classical approach. For instance, through feature selection we could find that the best feature is happiness; if this is the case we would expect that the highest correlation is between the happiness feature and behavioral intentions. Indeed, if the best feature is happiness and has a positive correlation we would expect the particular behavioral intention to increase when happiness increases. The third item is to mimic the theory-driven approach with predictions about the temporal importance of the data. It is computed by taking the four features with the highest known correlation with the behavioral intentions and extending this with two feature transformations: averages of the first (discrete) derivative and the difference between the averages of the first and the second half of the temporal sequence.

Compared to the first two items, we expect to find even better performance gains of the learning algorithms on the feature transformations, as they were educated guesses on where to find the strongest influence on the behavioral intentions. However, we should keep in mind that we have applied a procedure to transform the data in such a way that they will be more useful to us than the raw data. Our goal is rather to apply a technique that can find the rule for the procedure for us. That is what we shall attempt when we get to the LSM. It may thus be reasonably expected the LSM at least matches the performance of these preselected feature transformations. When the LSM is used for regression, we mean that we run the SVR on the output of the liquid in the LSM (see Chapter 3 for more details). In this sense the regression output is directly comparable to the output when using the averages or hand-picked and transformed facial emotion features.

To summarize, we have selected three baseline levels, each of which should perform progressively better at predicting unseen examples by the machine learning algorithms. The LSM should at least improve on the averages and may be reasonably expected to be at the level of the preselected feature transformations. This way we will have demonstrated that the ML tools we introduce actually extract more information from raw data than statistical tests do, and can point us in the direction where to look for important effects in the data. By introducing the paradigm in which we predict parts of the data, we deepen our understanding of the data and can discover unexpected effects with the machine learning algorithms.

2.5 Supervised Learning Algorithms

We are trying to find the best model with which we can predict the behavioral intention of a participant. This comes down to finding the best approximation of the following function:

$$f(\mathbf{x}^i) = r^i \tag{2.4}$$

where $\mathbf{x}^i$ is a feature vector of participant i and r^i is the target value for that participant. The feature vector can, for instance, contain the averages of facial features over the whole video or features produced by some other transformation. The output is a single numerical value, the behavioral intention of the participant, which should be as close as possible to the target value r^i. The task of the following algorithms is to provide a function to best estimate r^i from a participant's set of features.

2.5.1 Pseudoinverse

If we simply wish to make a multivariate regression which best explains how to fit the examples we can do so according to the equation:

$$f(\mathbf{x}^i) = \mathbf{w}^T\mathbf{x}^i + w_0 \tag{2.5}$$

where $\mathbf{x}^i$ is a feature vector of participant i and $\mathbf{w}$ is a vector with weights for each feature. For this we can actually use the pseudoinverse to gain an analytical least squares solution. If we have a matrix $\mathbf{X}$ with on each row a participant from the training set and in each column a corresponding feature, we wish to multiply it by an optimal weight vector $\mathbf{w}$ (now the feature vector and weight vector include the bias term w_0 for notational convenience) of the length of the features to give the target values of all participants $\mathbf{r}$. In other words, we are looking for a solution of the following system:

$$\mathbf{X}\mathbf{w} = \mathbf{r} \tag{2.6}$$

The solution is simply:

$$\mathbf{w} = \mathbf{r}\mathbf{X}^+ \tag{2.7}$$

where $\mathbf{X}^+$ is the Moore-Penrose pseudoinverse. We can then use the weight vector $\mathbf{w}$ to predict the responses of the test set. The MSE will determine the performance of the regression. This solution gives a linear regression, which may not be powerful enough for the difficult data we have at hand, so we introduce powerful non-linear alternatives in the next sections. As will be discussed in Chapter 3, the recurrent neural net in the LSM casts the input data into a highly non-linear space, so perhaps then the transformed input can be separated by this linear regression.

2.5.2 Artificial Neural Networks

The artificial neural network (ANN) is an extension of the perceptron, which has a nearly identical form to the linear regression above which could be solved with the pseudoinverse. However, in an ANN, there are multiple layers of features. In the layers between the input and output layers, the so-called hidden layers, non-linear transformations can be introduced, resulting in a non-linear model. The architecture of a simple neural network with one hidden layer is demonstrated in figure 2.4. The input layer receives the normalized facial feature

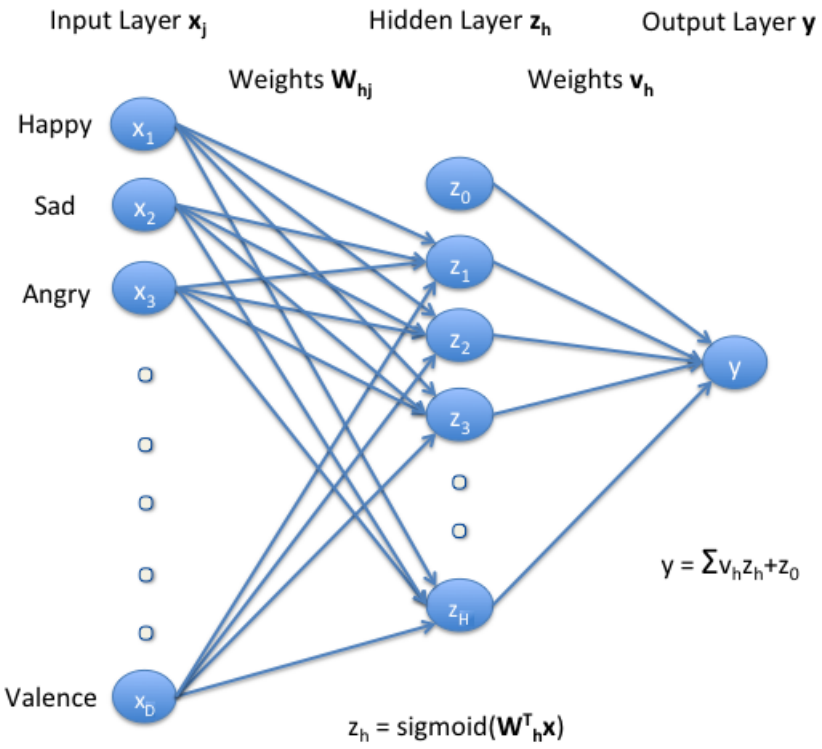


Figure 2.4: Artificial neural network with one hidden layer

vector from our data set. In this example, it could be the averages of the facial emotions that were presented as features to the ANN in one long vector, losing the temporal aspect of the data. The input nodes are connected to the hidden layer through an initially randomly

generated set of weights. In the hidden layer the activation of the input features is summed and a non-linear transformation is applied to them. On the input $\mathbf{x}^i$ for a participant i, we calculate the activation of the hidden layer (following Alpaydin, 2010):

$$z_h^i = sigmoid(\mathbf{W}_h^T \mathbf{x}) = \frac{1}{1 + exp[-(\sum_{j=1}^{D} w_{hj} x_j^i + w_{h0})]} \tag{2.8}$$

with $\mathbf{W}_h^T$ the matrix of weights between the input and hidden layer for the H hidden units and D input units. The bias w_{h0} on the right hand side is already included in $\mathbf{W}_h^T$. In turn, the hidden layer is connected to the output node. Here, we have only one output node, as this represents the behavioral intention. The output is calculated as follows:

$$y^i = \sum_{h=1}^{H} v_h z_h^i + z_0 \tag{2.9}$$

with $\mathbf{v}$ the weights between the hidden and output layers, which are also randomly initialized. At first, the output will not make any sense, as the weights in the network are all randomly generated. Therefore, the network needs to be trained. The error of the output with respect to the target value was calculated using the squared error term $E = \sum_i \frac{1}{2}(r^i - y^i)^2$, where r^i is the actual target value and y^i is the estimated activation for some example i. As we know this error, we can adjust the weights of the previous layer with the derivative of the error in such a way that for the same input, the output error would be reduced. We get the following update of $\mathbf{v}$

$$v_h = v_h + \eta \sum_i (r^i - y^i) z_h^i \tag{2.10}$$

The learning rate η was initially taken as 0.05, but updated later in cross-validation. Notice that the pattern of the update is $learningfactor \times (desiredoutput - actualoutput) \times input$, moving any future output on the same input closer to the desired input. This is the equation

for batch gradient descent where the weights are updated with the sum of the error over all examples (participants). It is also possible to perform online updates (stochastic gradient descent) where the weights are changed after each example i. Finally, in order to adjust the weights between the input and the hidden layer, the error correction is backpropagated to the weights of $\mathbf{W}$:

$$\begin{aligned} w_{hj} &= w_{hj} - \eta \frac{\partial E}{\partial w_{hj}} \\ &= w_{hj} - \eta \sum_i \frac{\partial E}{\partial y^i} \frac{\partial y^i}{\partial z_h^i} \frac{\partial z_h^i}{\partial w_{hj}} \\ &= w_{hj} + \eta \sum_i (r^i - y^i) v_h z_h^i (1 - z_h^i) x_j^i \end{aligned} \tag{2.11}$$

Again following Alpaydin (2010). As the desired output and hence the error of the hidden layers is not known, the chain rule is used to backpropagate the error through $(r^i - y^i)v_h$, where the involvement of the hidden layer through v_h is taken into account. The remainder of the update contains the derivative of the sigmoid and the input term, so that again we have the ingredients to compare the desired output with the actual output given a certain input and change the weights accordingly. By performing gradient descent, the weights are gradually altered each epoch (training iteration) until convergence to a minimum was achieved or 500 epochs had passed, after which the accuracy practically did not improve anymore. However, there is overfitting of the data at such a high accuracy, which means that the algorithm has learned specific idiosyncrasies of the examples in the training set, leading to poor generalization in the test set. For this reason, every 20 epochs the performance on the validation set was also assessed and the weights of the ANN were saved for the best validation set. Finally, the accuracy on the test set was determined, which is presented in the results section. This idea of saving the best weight configuration is known as early stopping, where the values for the weights of the NN are used from an earlier point than the final epoch, in order to avoid overfitting.

We briefly experimented with a two hidden layer ANN as well, but this did not improve

results over using just one layer. The hidden layer had between 2 and 20 hidden units and the output layer was one node producing the regression result. We implemented the algorithm in R (R Core Team, 2013), using matrix multiplication whenever possible.

2.5.3 Support Vector Regression

The support vector machine (SVM, see Cortes and Vapnik, 1995, for the popular soft margin version) is often used as a tool for classification. The main feature of an SVM is that it constructs an optimal hyperplane that separates classes with the largest margin, using support vectors to demarcate the boundaries. As it often happens that linear separation is not possible in the input space, this space is mapped to a higher dimensional space, in order to make separation possible. To keep the computational load reasonable, SVMs are designed in such a way that dot products are computed in the original space, but according to a kernel function. This mechanism alleviates the necessity for a transformation of all samples into the higher dimensional space, while we are still being able to estimate decision boundaries therein. It was shown that the same principle can be applied to regression (Vapnik, 2000).

First, it must be noted that in SVR, the ϵ-sensitive loss function is used instead of the MSE:

$$E_\epsilon(r^i, f(\mathbf{x}^i)) = \begin{cases} 0 & if \quad |r^i - f(\mathbf{x}^i)| < \epsilon \\ |r^i - f(\mathbf{x}^i)| - \epsilon & otherwise \end{cases} \tag{2.12}$$

It can be seen from the equation that errors up to ϵ are not considered as an error. Moreover, the error function is linear outside the ϵ zone. Together these aspects make that this error function is more tolerant toward noise (Alpaydin, 2010). The ϵ is a hyperparameter that will be optimized during cross-validation.

Let $\mathbf{x}^i$ be the input vector and r^i be the target value of example i. Our goal is to find the hyperplane that maximizes the margin over all regression examples. A hyperplane consists

of all $\mathbf{x}$ satisfying $\mathbf{w}^T\mathbf{x}^t + w_0 = 0$. However, it is not feasible that such a function exists for most data sets, so we need to introduce slack variables ξ_i, ξ_i^* which allow for optimization while allowing error outside the ϵ zone. The problem can be formulated as an optimization problem:

$$\begin{aligned} &minimize && \frac{1}{2}||\mathbf{w}||^2 + C\sum_t(\xi_i + \xi_i^*) \\ &subject\ \ to && \begin{cases} r^i - (\mathbf{w}^t\mathbf{x} + w_0) \leq \epsilon + \xi_i \\ (\mathbf{w}^t\mathbf{x} + w_0) - r^i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{aligned} \tag{2.13}$$

An example for the linear case we are discussing is given in Figure 2.5. The C constant is the next hyperparameter we set in cross-validation, it specifies the tradeoff between the flatness (how small the norm $||w||$ is) of the function $f(\cdot)$ and the extent to which deviations outside the ϵ zone are allowed. If C is very high, the function will fit the training data very well but may be less generalizable. If C is very low, $f(\cdot)$ will be very flat, and similar outputs will be produced for every input (for instance close to the mean).

The solution of the optimization problem is calculated using the dual formulation, where

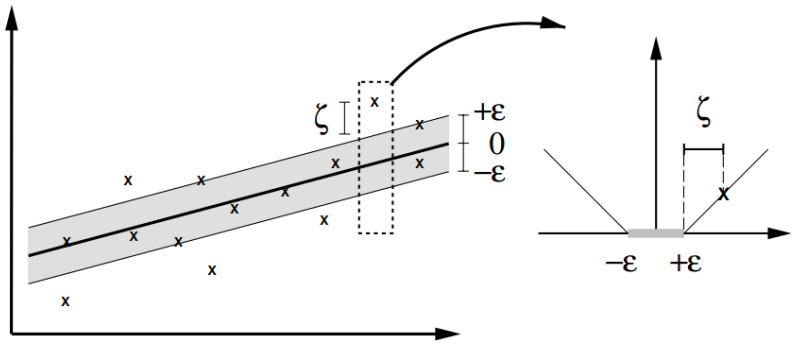


Figure 2.5: linear support vector regression with soft margins, from Smola and Schölkopf (2004)

the optimum is given by the saddle point of the Lagrange function. Smola and Schölkopf (2004) offer full details; we refrain from restating lengthy formulas. For the purposes of gaining insight in optimizing the hyperparameters it suffices to state that the solution of the function $f(\cdot)$ can be formulated as follows:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \sum_i (\alpha_i - \alpha_i^*)(\mathbf{x}^i)^T\mathbf{x} + w_0 \tag{2.14}$$

with constraints $\sum_i(\alpha^i - \alpha_i^*) = 0$ and $0 \leq \alpha_i, \alpha_i^* \leq C$. Now, $(\mathbf{x}^i)^T\mathbf{x}$ can be replaced by a kernel $K(\mathbf{x}^i, \mathbf{x})$ to achieve a non-linear regression fit. Note that the kernel is applied directly in the input space. We use cross-validation to find which kernel works best to classify our data. Apart from the linear kernel, we test the sigmoid kernel:

$$K(\mathbf{x}^t, \mathbf{x}) = tanh(-\gamma\mathbf{x}^T\mathbf{x}^t + 1) \tag{2.15}$$

and the Radial Basis Function (RBF) kernel:

$$K(\mathbf{x}^t, \mathbf{x}) = exp(-\gamma||\mathbf{x}^t - \mathbf{x}||^2) \tag{2.16}$$

in which γ is the kernel parameter. This is the final parameter we need to optimize. For the RBF kernel, it determines the relative size of a bell-shaped surface around each support vector. If the surface is too small, there will be overfitting and if the surface is too large the approximation of the function may be too biased. For the sigmoid kernel, γ influences the steepness of the *tanh* function. The tolerance of the stopping criterion could also be set, but after brief experimentations it was held at the default value.

It was not in the scope of this project to make an implementation of support vector regression. Instead, we opted for the popular libSVM (Chang and Lin, 2011) implementation,

which has been interfaced for *R*. In order to assess which parameters fitted the data best, we used sixfold cross-validation. This means that the training set was split into six equal parts, after which a procedure is run six times where one part is left out as a validation set and the remaining parts are used for training. So each data point in the training has then been used in both training and validation. Finally, the MSE result on the test set is determined by comparing the actual value with a prediction from a model trained on the entire training set with the best parameters.

2.6 Feature Selection

From the behavioral data, we would like to find out which of the emotions deduced from facial expressions are important for understanding behavioral intentions. For the purposes of interpretation, we could not reduce dimensionality by extracting better features from existing features. Instead, we applied feature selection wrapped around the whole training and testing procedure: one feature is selected and omitted after which the whole system is trained and tested. If performance deteriorates severely after a feature is omitted, the feature must have been useful for the regression. Conversely, if the performance is unaltered or even increased, the feature was most likely not useful. We repeat the procedure for each feature and after all features have been left out once, it is determined which feature had the least impact on the performance. This is called backward feature selection and it was applied to leave out the 6 least useful features. Afterwards the remaining 4 features were trained and tested in isolation and in every possible combination with the other features (i.e. $\binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 15$ combinations were tested), resulting in a list of which features and combinations of the 4 best features predict the behavioral intentions best. The following pseudocode further elucidates the process:

Algorithm 1 Feature Selection

$A \Leftarrow \{1:10\}$ {A is initially the set of 10 features}
while $numberOfElements(A) > 4$ **do**
  $N \Leftarrow numberOfElements(A)$
  $M \Leftarrow N - 1$
  $Results \Leftarrow Array(Length(N))$
  loop $\{Ntimes\}$
    $a \Leftarrow A[N - M]$
    $B = A/a$
    $M \Leftarrow M - 1$
    $model \Leftarrow trainLSM(B)$
    $Result[N - M] \Leftarrow testLSM(model)$
  end loop
  $A \Leftarrow A/A[which(max(Results))]$
end while
{Train and test all different remaining feature combinations}
$Separate : \{1, 2, 3, 4\}$
$Combinations : \{12, 13, 14, 23, 24, 34\}$
$Combinations : \{123, 124, 134, 234\}$
$Complete : \{1234\}$

2.7 Summary

In this chapter, we have discussed the paradigm for testing predictions from a machine learning perspective and applied that perspective to a problem from behavioral marketing. Briefly, we discussed how the analysis would normally be performed and then suggested that we should create models to predict the dependent variable. This methodology could bring to light previously unexpected effects in the data. We surveyed popular non-linear supervised learning algorithms that create models to predict complex behavioral intentions from the facial expressions of participants who watched a commercial. We can even find which facial expressions are particularly important by applying feature selection.

However, so far we have simplified the data by transforming them into a representation without s time dimension. We chose three simplified approaches to predict the behavioral intentions: the average behavioral intentions of the test set (which should be the worst that an algorithm can model), the facial emotions averaged over the sequence of the commercial and a theory-driven selection of the best features transformed so that it implicitly includes

temporal information. We will run the supervised learning algorithms on these approaches to set baseline results. Each approach is expected to improve over the result of the last.

In the following chapter we will describe how we intend to extract more information from the temporal dimension of the facial features. Essentially, we will train a recurrent neural network to recognize temporal patterns in higher order combinations of features. We expect that we should be able to match and hopefully improve upon the baseline that had the theory driven selection and transformation of features.

3 Liquid State Machine

In this chapter, we will extend the analysis of temporally complex behavioral data by introducing the Liquid State Machine (LSM), a machine learning algorithm that is well suited for analyzing temporal data. We have seen that we can apply supervised learning in order to train models that could predict a participant's behavioral intentions after having seen a simplified (i.e. non-temporal) representation of their facial emotions. However, these representations of the emotions were only a crude estimation over the entire temporal sequence. Here, we will explain how the LSM extracts complex representations that take temporal patterns into account. The general theory of Reservoir Computing (RC) and Echo State Networks (ESNs) will be concisely introduced.

Unfortunately, the LSM currently requires extensive parameter optimization in order to perform well. For the same parameter initializations, generated recurrent neural networks can behave quite differently. We will discuss approaches to minimize the optimization time and to find the on average best performing recurrent neural nets. We suggest multiple network topologies that may improve the performance of the LSM on our data. Finally, it will be discussed how we can make explicit use of the temporal dimension of the LSM.

3.1 Recurrent Neural Networks

To analyze a temporal sequence in the artificial neural network paradigm that we have introduced, it was originally proposed that recurrent connections should be added to the network architecture. The concept is straightforward: the neurons of a hidden layer should be activated by nodes other than the input. Instead of a feedforward structure, there is also feedback from hidden neurons to other hidden neurons or to themselves. The input that

hidden neurons receive from each other depends on their activation at previous steps in time. In other words, with each step in time, a hidden "layer"[1] is activated directly by the input of that time step and simultaneously by recurrent connections of context nodes that contain information from previous steps in time.

We can state the latter concepts formally by considering the current state of a recurrent neural network as the time dependent state vector $\mathbf{x}(n) \in \mathbb{R}^{N_x}$, with N_x the number of hidden units. This vector is updated at time step n by a function which depends on the previous state of the network and new inputs from the input vector $\mathbf{u}(n)$, thus leading to a recursive definition (cf. Lukoševičius and Jaeger, 2009):

$$\mathbf{x}(n) = f(\mathbf{x}(n-1), \mathbf{u}(n)) \tag{3.1}$$

Where the function $f(\cdot)$ is the activation function and depends on the paradigm that is used. We will discuss this in detail in the ESN and LSM sections. More concretely, the input matrix that distributes the input activation over the network is given by $\mathbf{W}_{in} \in \mathbb{R}^{N_x \times N_u}$, where N_u is the input dimension, and the connections of the liquid are $\mathbf{W} \in \mathbb{R}^{N_x \times N_x}$. These weight matrices result in the following update:

$$\mathbf{x}(n) = f(\mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}\mathbf{x}(n-1)), \qquad n = 1, \ldots, T \tag{3.2}$$

In this formulation, $\mathbf{W}$ represents the connectivity and the topology of the network. The output (i.e. the behavioral intention in our case) can be calculated as follows:

$$\mathbf{y}(n) = f_{out}(\mathbf{x}(n)) \tag{3.3}$$

where $f_{out}(\cdot)$ is the output function, which in the case of linear regression could be the weight

[1]The term layer becomes redundant as the topology has become more complex through the recurrent connections.

vector $\mathbf{w}_{out} \in \mathbb{R}^{N_x}$ which makes a linear combination of all the nodes in the state vector to output a regression value. Moreover, n in $\mathbf{y}(n)$ is redundant in our problem, as we output one expected value for the behavioral intention over the whole sequence of a commercial.

Early examples of recurrent neural network architectures which had context nodes (that were not influenced by the input) include that due to the work of Elman (1990). Even earlier it was found that a network with recurrent connections could be trained with back propagation (Pineda, 1987). This was done by unfolding the network over time: the error can be back propagated through recurrent connections by treating each time step as an extra layer in a feedforward network. However, the main problem is the problem of the vanishing gradient, the effect that error becomes progressively smaller the further it is back propagated (Hochreiter et al., 2001). As such, the number of epochs that is required to converge to a solution is increased, if it will be feasible at all to converge. In addition, back propagation is costly and thus difficult to perform over many layers. Lastly, whereas back propagation has a clear interpretation in feed forward neural networks, it is not clear whether back propagation through time is the optimal way to obtain the best predictions over entire sequences.

The costliness has been reduced in recent years by the introduction of hessian-free optimization (Martens and Sutskever, 2011). Moreover, Hochreiter and Schmidhuber (1997) have proposed a method that circumvents the problem of the vanishing gradient: the long short-term memory (LSTM). The latter contains blocks that can effectively trap the error inside a memory unit and are thereby able to learn arbitrarily long temporal relationships, preventing the need to roll out the network during back propagation. LSTM has proven successful in speech recognition (Graves et al., 2013) and handwriting recognition (Graves and Schmidhuber, 2009), among other fields. In the following section, a paradigm will be introduced that circumvents both costly training and the vanishing gradient.

3.2 Reservoir Computing

In two different fields independently and roughly simultaneously came propositions to avoid the training problems in recurrent neural networks. Both suggested not training the recurrent

connections; rather, only the readout had to be trained. Though it sounds counterintuitive, it turns out that the cycles in an RNN topology make it a dynamical system, sufficiently complex to have a dynamical memory without explicit training of the recurrent connections (Lukoševičius and Jaeger, 2009). In the field of computational neuroscience, the Liquid State Machine was introduced (Maass et al., 2002), whereas in machine learning Echo State Networks were presented (Jaeger, 2001). We will first discuss the echo state networks as their mathematical description has fewer parameters and allows us to express the LSM in terms of an elaborated ESN approach which includes spiking neural activations.

The main difference in the approaches is their activation model for the neurons. The activations of the neurons at any time step n are described in the state vector. The ESN works with discrete time steps and has a sigmoidal activation function, therefore the state vector contains real values at each step in time. The LSM state vector contains binary coding similar to the biological model from which it originates: a neuron in the network has either fired or it has not. In order to determine whether a neuron fires a thresholded activation function is used. Each neuron has a potential which is influenced by the input and by other neurons that have fired in the previous time step; if the accumulated potentials reach a threshold, the neuron fires and propagates its signal to other neurons to which it is connected in the next time step.

Despite these differences, the algorithms share the same theory: a significantly informative representation of a time-dependent classification or regression problem can be derived from a complex and sensitive enough dynamical system. Instead of training a recurrent neural network we generate one which has high enough complexity to cast the input onto a higher dimensional space in which the temporal component is implicitly represented. In the next sections we will discuss the *echo state*-, *approximation* -, and *separation* properties which are necessary for the system to behave in this way. However, due to the lack of training we can already recognize the main problem, namely that there has to be performed a large amount of (meta)parameter setting in order to let the recurrent nets behave well. As the theory has not been completely developed yet (if this is even possible), there is no one size fits all solution for generating a recurrent neural net that works in each situation. Inputs have to

be carefully scaled and sometimes several networks have to be generated to discover which dynamical system works best for the problem at hand.

We should realize that the reservoir paradigm was initially used for time series prediction or recognition where there is an interpretable output after each time step. However, in the behavioral marketing problem at hand, we have a complex temporal sequence as input, but the output is just one single value for the whole time sequence. Thus, we will have to come up with a metric that is suitable for this type of regression. From the LSM we will extract a rate coding from the reservoir at several points in time which we can then use as input to our regression algorithm.

3.2.1 Echo State Machine

In this section, we will discuss the equations of the echo state network relevant to our problem and offer advice about the optimization of the recurrent neural network in this structure. We will first fill in the activation function, which is a sigmoid function in the case of the echo state network:

$$\tilde{\mathbf{x}}(n) = tanh(\mathbf{W}_{in}[1; \mathbf{u}(n)] + \mathbf{W}\mathbf{x}(n-1)) \tag{3.4}$$

We have added the bias node 1 to the input explicitly. As can be seen from the update equation, at each step in time the activations from the previous step in time from all connected neurons will influence each other. For echo state networks it is standard procedure to include the input at each time step when predicting the output:

$$\mathbf{y}(n) = f_{out}(\mathbf{W}_{out}[\mathbf{u}(n)|\mathbf{x}(n)]) \tag{3.5}$$

Here the $[\cdot|\cdot]$ indicates that the input and state vectors are concatenated.

The network discussed so far is based on sigmoid units which only fractionally and indirectly depend on their previous value, making it difficult for them to learn slow dynamics

(Jaeger, 2001). Therefore, it is more appropriate to use a leaky integrator firing model, approximating real time effects while still using discrete time steps (it is sometimes called a hybrid system for this reason). This brings the echo state network closer to a neurobiologically plausible system, bearing similarities to the LSM. The update of the state is done according to the following update equation, where α indicates the leaking rate.

$$\mathbf{x}(n) = (1 - \alpha)\mathbf{x}(n - 1) + \alpha\tilde{\mathbf{x}}(n) \tag{3.6}$$

The leaking rate effects the speed of the dynamics and is one of the parameters that has to be optimized for each problem, as the speed of the dynamics of the input can vary widely.

But now that we know how the echo state network works, we still have to answer why it works. What are the conditions that have to be met in order to learn from a recurrent neural network that has not been trained? First, the network $\mathbf{W}$ has to be a large, sparsely connected matrix of which the connections are generated randomly. This way there are many different activations that will influence each other and thus lead to different activations as they are randomly connected. The second important property is the *echo state* property, which in essence dictates that for any past input $\mathbf{u}(n)$ and activation $\mathbf{x}(n)$, the effect of that input and activation should fade in the future, i.e. the past influence of past activation should vanish at $\mathbf{x}(n + k)$ as $k \to \infty$ (Jaeger, 2001; Lukoševičius and Jaeger, 2009). There is a great intuition to this property because if the input and past activation do not fade, it may actually amplify and influence later input to the extent that they are not separable anymore by an output learning mechanism. This is to say that the system may become chaotic and nothing can be learned from it anymore.

Depending on the input for a problem and the particular initial parameters, it is conceivable that one generated liquid may act chaotically whereas another may not. Therefore, it is important to see if there is a way to generate or tune a network in such a way that it will not become chaotic but still performs a significant transformation over the input. Unfortunately, there is no one clear algorithm which solves all the problems but there are some guidelines

from the theory. The echo state property has been shown to be obtained "for most practical purposes" (Jaeger, 2001) if the spectral radius (the largest absolute eigenvalue) of $\mathbf{W}$ is smaller than 1, i.e. $\rho(\mathbf{W}) < 1$. However, it is not a mathematical guarantee, the only guarantee is that the echo state property does not hold if $\rho(\mathbf{W}) > 1$ in a system with the *tanh* activation function for the neurons and zero input. On the positive side, the property can still be obtained if $\rho(\mathbf{W}) > 1$ for non-zero input and indeed the best performance has often been found experimentally with much larger spectral radii. It can intuitively be seen that the spectral radius will need to be tuned to the amount of memory and non-linearity needed given an input. There will be more non-linearity if the spectral radius increases because neurons with recurrent connections will have a stronger influence on each other and thus the topology of the network will have a stronger influence on the state of the network through time. This will also increase the memory potential, at least if the activation does not become chaotic and the echo state property is lost. A mathematical proof of the conditions for the echo state property to hold can be found in Jaeger (2001). In the next section, we will see how the theory of the LSM continues the line of thought described here about the ESN.

3.3 Liquid State Machine

Like the ESN, an LSM (Maass et al., 2002) is an elaborate construction that contains a recurrent neural network and is therefore very suitable for input that changes over time. The LSM temporarily retains the information that has already been presented to it. How much and how long it retains information depends on the parameters of the system. It takes input features from time slices and feeds them into an automatically generated recurrent neural network, the liquid, which acts as a dynamical memory, "able to process temporal context information" (Lukoševičius and Jaeger, 2009). The weights in the liquid are not trained after they are generated and only the liquid's internal state changes over time when it is excited by the input. The input is fed into the liquid after which the current state of the liquid is updated according to its previous state and the activation dynamics. The learning part of the Liquid State Machine takes place after the liquid is excited, as the state of the liquid is taken as input to an SVR in our case, see figure 3.1.

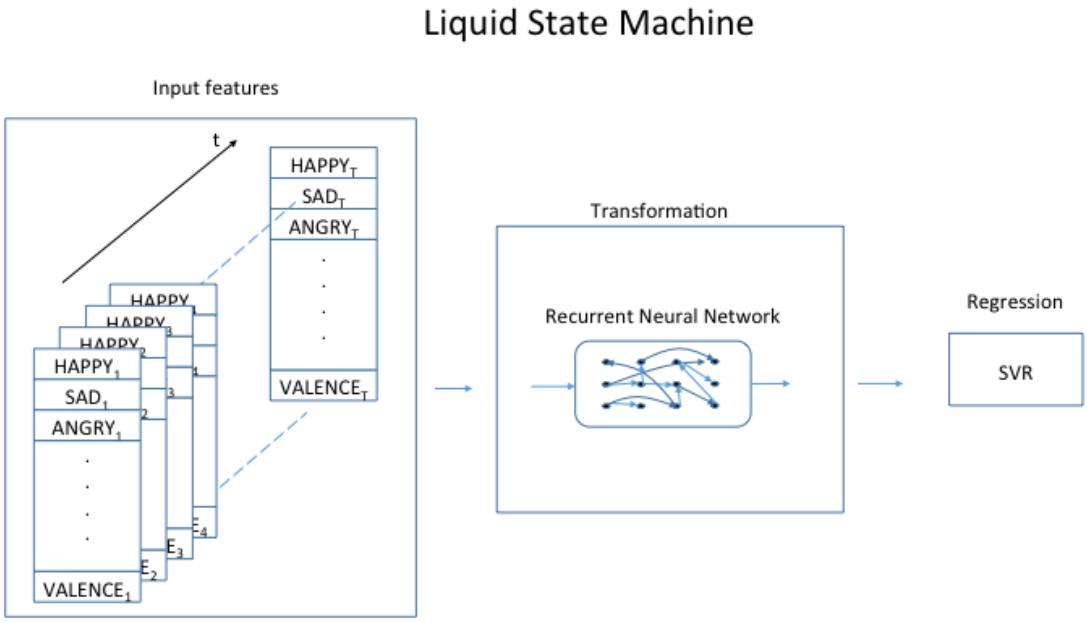


Figure 3.1: Schematic of the Liquid State Machine elucidating the two main transformations that happen in the LSM. First the input excites the liquid, a recurrent neural network that acts as a dynamical system casting the input into a higher dimensional space. Second the SVR that can transform a rate coding of the liquid into an output for a regression problem.

The activation equations are in principle the same as stated in equation 3.1, although in the literature the state of a liquid state machine M is said to be based on the mapping of a filter L^M (the liquid) from the input $u(\cdot)$. So the state vector is expressed more generically than equation 3.1 as:

$$\mathbf{x}(n) = (L^M u)(n) \tag{3.7}$$

The main conceptual difference is that the input function $u(\cdot)$ can be any continuous function. The output function $y(\cdot)$ is identical in formulation, but here, too, this could in theory be any continuous function. The result is the same: the current state of a liquid depends on the input and the previous state of the liquid, only the formulation is neutral toward the implementation that connects different states of the liquid. Our implementation of how the liquid is generated and activated will be discussed in the next section.

The approach taken to explain why the LSM actually works without training its weights is slightly different than before. The LSM literature defines two essential properties: the *separation* property and the *approximation* property. For two different inputs there will be two different trajectories of internal states; the separation property refers to the amount of separation between these trajectories. The approximation property refers to the ability of an LSM to transform the internal states of the liquid into the desired output. It was mathematically demonstrated in Maass et al. (2002) that if both properties are met, no matter what implementation is used, an LSM has universal computational power. Similarly, it was demonstrated that spiking neural networks in this context can have universal computational power, which is an extraordinary result as we realize that any continuous function in time can be captured by a brain-like structure. However elegant this result, it is by no means trivial to find out when and if the separation and approximation properties are met and how to produce a system that can solve a problem adequately and within reasonable time. As we are applying the LSM to a real life problem in this thesis, these are the answers we need to find.

In the section on the ESN, we saw that the network need not be trained in part due to the *echo state* property. In LSM theory, there is a similar property called *fading memory*, which states that "the most significant bits of its current output value $F(u)(0)$ depend just on the most significant bits of the values of its input function $u(\cdot)$ in some finite time interval $[-T, 0]$ going back into the past." This property ties in with the idea that the liquid must not be chaotic, otherwise the most significant bits of a state at any given time may be dictated by the structure and activation of the dynamical system, instead of the input function.

Learning is done by training an SVR on the rate coding from the liquid. The rate coding is simply the amount of times a neuron has fired over a certain period of time. The dynamics of a liquid are represented by the amount neurons fire. After 10% of facial features responses from the commercial have excited the liquid, we calculate the rate coding over that particular 10%. The first 10% is seen as a warmup phase for the liquid, but thereafter the rate coding of each next 10% of the states of the liquid is used as input for SVR. The state of the liquid changes dynamically because of recurrent connections in the network and the output is therefore a "nonlinear transformation of the input history" (Lukoševičius and Jaeger, 2009).

The SVR is learning that these differing states actually belong to the same class. In the readout of the LSM, the model from SVR is used to predict the regression values given a new state of the liquid (i.e. a new participant). The MSE is calculated on the test set.

3.4 Generation and Activation of a Liquid

We have initially modeled our LSM after Pape et al. (2008). The liquid takes the input by multiplying a random uniformly generated input weight vector with the facial features. For the LSM we presented 10 features (6 basic emotions + contempt, neutral, valence and arousal) from every frame in turn to the liquid. The size of the liquid was initially taken as 100 neurons in a square format, so 10×10. Varying the size of the liquid demonstrated differing results; generally when using larger liquids the results were more consistent although training time did increase significantly.

For the generation of the connections of the liquid we used the following equation (from Maass et al. (2002); Pape et al. (2008)):

$$p_i^c(j) = Ce^{-(\frac{D(i,j)}{\lambda})^2} \tag{3.8}$$

where $D(i,j)$ is the Euclidean distance between point i and j. This equation gives the probability that point i is connected to point j, which is affected by the parameters C and λ. The values of the parameters are given in Table 3.1. The topologies of the liquid via this rule and other generation rules are discussed in detail in the next section.

The liquid is modeled as a discrete time spiking neural network. By this we mean that it is not the goal to interpret the input with a system that closely resembles a biological system on a millisecond scale, but rather to have a well-performing tool that includes computational elements from a real brain. In this sense, it is still different from the ESN, in that the neural network is spiking with neurons with a refractory period, meaning that it will be well suited for non-linear processing on long distance temporal patterns because it can have a long

memory. The features are introduced as additional membrane potentials spreading across the liquid, governed by the input weights. The potentials will then influence, and propagate throughout the system via the connections established in equation 3.8 above if a neuron potential exceeds a threshold α and fires.

There will be a refractory period in subsequent slices after a neuron has fired in which the potential is reduced to 0.1. In addition, a relative refractory period is used during which the potential can increase but only moderately. The spiking of a neuron $x_i(t)$ is simply 1 if the neuron has reached the threshold of the membrane potential (the most recent timestep of firing is indicated by t_i^f) and if it is not in the refractory period and 0.1 otherwise. So the state vector is basically a binary vector, but now we need an additional vector $\mathbf{m}$ to store the membrane potential. The membrane potential is calculated as follows:

$$\mathbf{m}(t) = \begin{cases} \tau\mathbf{m}(t-1) + \mathbf{W}\mathbf{x}(t-1) + \mathbf{W}_{in}\mathbf{u}(t) & if \quad (t - t_i^f) > t_i^a \\ 0.1 \quad otherwise \end{cases} \tag{3.9}$$

Here we also see the variable τ that indicates the leaking rate. It models the rate at which neurons lose potential through time. Again, though the implementation is discrete, some continuous aspects of the biological model are retained, making the LSM a suitable tool to approximate continuous functions in real time. The values that were inspected during parameter optimization are given in table 3.1. The implications from the optimization process will be discussed in the results and discussion sections.

Parameter Description	Symbol	Value
liquid dimensions		$(6:14)^2$
connectivity parameter	C	(0.4,0.6,0.8,0.9)
connectivity parameter	λ	(1.6,1.8,2.0,2.2,2.6,3.0)
proportion of inhibitory values	p^i	0.1
weight between unit i and j	w_{ij}	0.4
firing threshold	α	1
half-time of the membrane potential	t^h	4.5
absolute refractory period	t^a	4
readout interval	t^v	10%
warmup period	t^w	10%,50%

Table 3.1: LSM parameters

3.5 Liquid Architectures

As mentioned earlier, the liquid is not generally considered a layer, but is sometimes referred to as the reservoir. Although it is generated with rules that incorporate a certain amount of randomness, this does not mean that we cannot say anything about its structure. The structure is not a layer, but it does matter what kind of connections are made. In fact, the performance can be impacted drastically by choosing and fine-tuning different architectures for the liquid. How well a liquid is able to separate an input stream is determined by three aspects: size; which connections are formed; and how strongly these connections influence the dynamics of the liquid. The latter aspect can be optimized by introducing unsupervised learning rules that can potentially tune a network to the input data and make it more suitable to separate into a higher dimensional space. However, learning rules change the synaptic plasticity of the neurons (weights), not the connectivity itself. We will discuss learning in later sections but first we discuss how the connectivity can be generated.

For the LSM, a sparse network is generated according to a distribution rule. From this rule we can predict how one neuron is connected on average to all other neurons. We use several rules of construction in order to obtain different network topologies. Different topologies have various advantages and disadvantages considering their network properties and therefore process input data differently. In this section we explain different topologies and how they

may transform the input data. Nevertheless, we will have to inspect all the topologies and select the most suitable topology for our data by trial and error as we do not know the preferred structure of the problem space beforehand.

3.5.1 Waxman

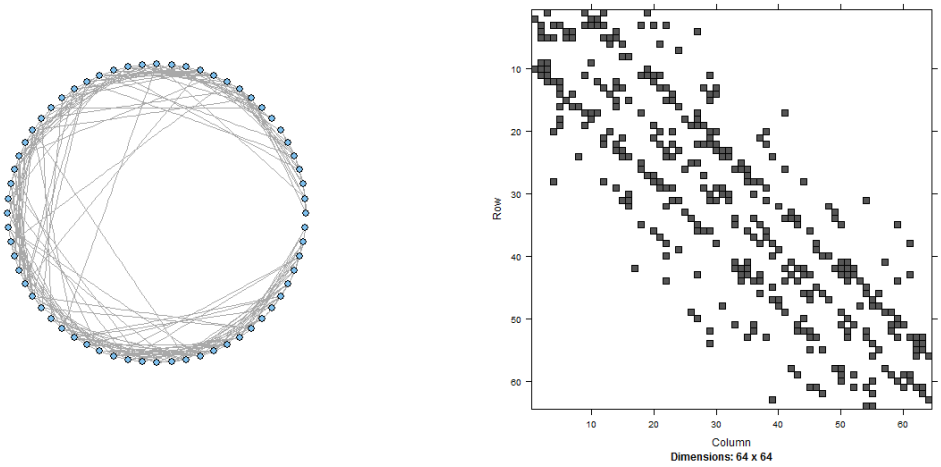
It should be noted that the architecture in section 3.4 is very much like that of the Waxman model (Waxman, 1988). The Waxman model is, in turn, a more complex version of a basic random network which connects each pair of nodes with probability p. The latter is often named the Erdös-Rényi (ER) graph, after the first researchers to model these random networks (Erdős and Rényi, 1960). The Waxman model bases the probability of two nodes having a connection on their proximity to one another:

$$p(i,j) = Ce^{-(\frac{D(i,j)}{\lambda})} \tag{3.10}$$

The probability of being connected falls exponentially with increased distance and is further influenced by parameters λ and C. C scales the entire probability and thus the number of connections made. λ scales the relative importance of the distance between two nodes. If λ is very large, there will be almost no influence and we are back with random connectivity from the ER graph. On the other hand, if λ is very small, it is nearly impossible to form long distance connections and we will only have local connectivity. A balanced structure with high local and sparse distant connectivity is a straightforward model of the real brain at first sight: the longer an axon is in the human brain, the more energy it requires to construct and to be used. In the abstract model, we can interpret the distance measure as the difference between two points in n dimensional Euclidean space. The parameters are cross-validated for C and for λ as indicated in Table 3.1. We used the formulation in equation 3.8 and two dimensional Euclidean space as the distance measure.

Figure 3.2 shows a typical Waxman architecture. For the distance measure, we use a two dimensional space, so the adjacency matrix shows that neighbors are also found at points

which would lie below a neuron in a square formation. At the far right side of the circle are the first and last neurons, and the circle can be followed in a counter clockwise fashion. It can be observed that this instance of the architecture has no very long distance connections but many close and medium length ones.



(a) Topology plotted on a circle (b) Adjacency matrix of the topology

Figure 3.2: A Waxman topology with $N = 64$, $C = 0.9$, $\lambda = 1.4$

3.5.2 Scale-Free

In a scale-free network all the connections for a neuron abide by a power law. Whereas the ER model (or the Waxman model for high values of λ) is expected to have the same average connectivity, most experimentally observed networks behave according to a power law. This means that the proportion of neurons $P(k)$, which has k connections is governed by:

$$P(k) \sim k^{-\gamma} \tag{3.11}$$

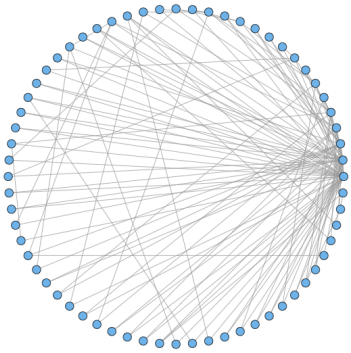
The distribution follows a power law, where very few nodes have high connectivity and many nodes have few connections. The network contains a few large hubs that connect most of the smaller hubs or separate nodes. It is clear that such a model will exhibit the small world phenomenon (Milgram, 1967), which is popularly known as the conjecture that any person in the world is connected by "six degrees of separation". The phenomenon dictates that the

shortest paths within the network are relatively short compared to the size of the network, as opposed to a network with a lattice structure where a node is only connected to some of its neighbors. However, even the randomly-generated ER graph will exhibit small world behavior, as will the Watts-Strogatz model, so this property holds for all network topologies we use for the liquid. The dynamics of the liquid will be different due to the distribution and, as we will see below, the clustering component.

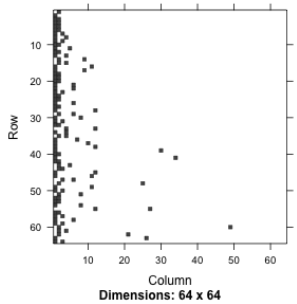
To create a scale-free topology for the liquid, we used the (Barabási-Albert (BA) model, see Barabási and Albert, 1999). This is a generating model with preferential attachment. Initially, there are m_0 nodes that are connected to each other. We can then view the generation process in time steps, where at each each time step t a node is added and is connected to each other node i (for a maximum of m connections) according to the following probability:

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j} \tag{3.12}$$

where k is the degree, the number of connections of that node i. The sum represents all the connections that have been generated so far. The equation shows that a new node will be more likely to connect to node i if it is a large hub with a high degree k as opposed to other nodes with just a few connections, which is clearly demonstrated in Figure 3.3. This generation process guarantees that the network will have a power law distribution (where $\gamma = 3$), which is in accordance with the experimental observations in biological networks (Bullmore and Sporns, 2009).



(a) Scale-Free topology plotted on a circle



(b) Adjacency matrix of the topology

Figure 3.3: A Scale-Free topology with $N = 64$, $m = m_0 = 2$

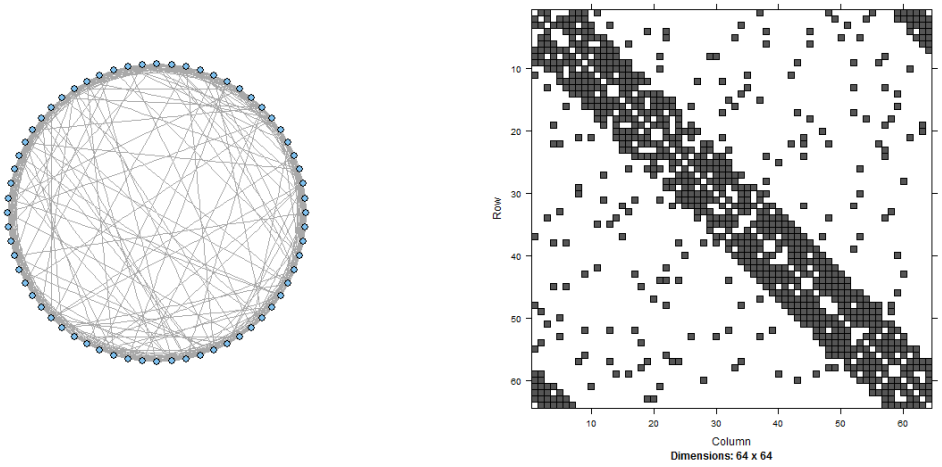
3.5.3 Watts-Strogatz

As our last topology, we examined the Watts-Strogatz (WS) small-world network model (Watts and Strogatz, 1998). This topology has been used as a model for the internet and other highly complex networks. We already found the scale-free BA model exhibited an important feature which has been experimentally observed. However, that model does not account for the clustering degree of a network. There are likely a few very large clusters but not many medium sized or smaller ones. Still, it has been experimentally observed that real networks often exhibit a clustering coefficient larger than that of the BA model. The Watts-Strogatz model uses a different way to generate a network that ensures more clustering.

First a lattice is constructed on a ring topology where every node is connected to its k nearest neighbors. Then every connection is visited and rewired randomly to another node with probability β. If $\beta = 0$ then there will be no rewiring and the network will have the lattice topology which has very high clustering, but does not exhibit the small world phenomenon. If $\beta = 1$ we would have the random topology similar to the ER model. The latter exhibits the small world phenomenon but does not have a realistic clustering coefficient. So by tweaking the parameters k and β we should be able to get a realistic network which

may have very useful dynamics for our behavioral marketing task. A typical example is found in Figure 3.4.

The only downside to this model is that it does not have the power law distribution which we saw with the scale-free network. Neither the BA nor the WS model is perfect in terms of how they mimic a natural network, but they each have an important aspect that has been experimentally observed. The Waxman model will have some degree of clustering and may be fairly similar to the WS model, though it is generated differently. Depending on the settings of the parameters, the Waxman model may have clusters which are close but not necessarily connected to the nearest neighbors. To examine how well a liquid performs with these topologies, we will experiment with all three models optimizing their parameters to find which produce the best performance on our data set.



(a) Watts-Strogatz topology plotted on circle (b) Adjacency matrix of the topology

Figure 3.4: A Watts-Strogatz topology with $N = 64$, $k = 14$, $p = 0.1$

3.6 The Edge of Chaos and Learning in the Liquid

It is thought that in order to achieve significant computations in an LSM the liquid needs to be close to displaying chaotic behavior, otherwise the input may dominate the pattern of the liquid and no interesting separation of the states occurs (e.g. Natschläger et al., 2005). However, if the liquid is chaotic, the internal dynamics of the liquid will obscure the input

and make the model useless for computations. There may be enough separation if the liquid is chaotic, but then there can be no approximation, so desired output cannot be achieved. The reflection can be found in the parameter search: if a liquid has low connectivity, not many non-linear effects can be learned, and if the connectivity is too high, no pattern can be discovered by the regression algorithm. However, for one set of parameters, it can take several (sometimes many) attempts of generating a liquid in order to find one that performs suitable computations. Therefore, it seems desirable to analytically obtain parameter settings that will generate liquids that operate in the critical regime. Moreover, benefit could be found by having a learning or scaling rule that can improve or fine-tune the connection weights after generating a liquid with a given parameter setting.

Several directions of theoretical research have been pursued. Legenstein and Maass (2007) estimate when the edge of chaos is reached by calculating the lyapunov exponent. This exponent indicates if, and how fast two state trajectories converge some fixed time after a given input. In another approach for spiking neural networks, the hamming distance is measured between the binary coding of the neurons (Natschläger et al., 2005). Although there may be a large dynamic memory at the edge of chaos, it is not entirely clear why the most efficient computations should occur there, or for which parameters. In the mentioned approaches, a connectivity parameter λ is explored, together with scaling of the weights of the liquid, e.g. the connectivity matrix $\mathbf{W}$, or scaling the firing threshold. Several "optimal" (combinations of) values can be obtained analytically for each connectivity parameter and size of the liquids using the methods described before, however, it is not clear which are to be selected over others. Moreover, these values are used by calculating the separation after giving different inputs and inspecting the state of the liquid at some later time.

This approach is definitely legitimate for clearly confined (theoretical) problems for which proof of concepts are important, but it is not clear how we should mimic this approach for our real-world problem. For example, even though we know the behavioral intention of a participant, we do not know when the relevant facial expressions take place that are uniquely representative for that intention. In other words, we do not know which input should separate the (future) trajectories to adhere to the separation property. If we use such a measure, we

do not know if we are evenly comparing states of the liquid. The best we can do is train the SVR and simply inspect the performance on a test set after we have learned some training set. Also, apart from synaptic scaling, the analytically obtained indicators for the edge of chaos only work for tuning the parameters of a hard-coded part of the generation of a liquid. They do not take into account one of the basic properties found in the human brain: synaptic plasticity.

Another way to estimate the separation property is to inspect the rank of a matrix that contains the resulting states of several input sequences that belong to different regression results (Legenstein and Maass, 2007). The theory ties in very elegantly with the theory of Vapnik-Chervonenkis (VC) dimensions. The VC dimension indicates the complexity of a learning algorithm by stating how many examples can be scattered (differentiated from each other). Simply put, if this number is too low, it will be impossible to separate different classes/regression values, but if it is too high (close to the number of training examples), the training instances can simply be stored and the algorithm will not be able to generalize well. If the rank of the matrix with the input examples is equal to that many examples, these examples can be linearly separated. However, if we take the rank of the resulting liquid states over the input of an entire training set we would like to see that the rank is much smaller.

Again we could ask how we can actually apply this idea to our data. For a controlled classification problem with few classes the VC dimension can offer a very good indication of performance. Sadly, we stumble upon the same problem as before: we do not know a priori which states are representative for a given regression value. It would go too far to expect that the facial responses at the beginning of a video are representative for a behavioral intention. If we find the optimum time for best separation, we have already optimized the parameters enough to find a good working liquid, so then we have already solved the main part of our problem. This leads to the predicament that we have influenced the problem space to the extent that it is unclear which conclusions can be drawn at the hand of the original data. The problem is not found with this line of research in general, but it is for our real-world problem where we have no a priori knowledge about the structure of the input function and have only one regression value over a whole time series.

There have been multiple attempts to set up synaptic scaling rules that will achieve dynamics that are close to chaotic behavior(Gómez et al., 2008; Natschläger et al., 2005). However, as we argued, for our data we can not establish clear reference points that represent certain regression responses, making it hard to apply sliding thresholds for instance as we may be over correcting for states that are too similar even when they belong to different categories.

Luckily, there are brain inspired techniques that can be applied to our spiking neural network that should tune the network for the input without necessarily moving it closer to the edge of chaos. There are three main synaptic plasticity rules that could be beneficial for our network. From Abbott and Nelson (2000) we find these three basic rules: synaptic scaling (also called synaptic normalization), spike-timing dependent plasticity (STDP) (also see Song et al., 2000) and synaptic redistribution (also known as intrinsic plasticity).

Applying STDP by itself has not proven very successful in the past. Partly this may be due to the fact that the weights of the synapses may grown unbounded. However, synaptic scaling can be used to keep the afferent connections of a neuron in balance. Intrinsic plasticity adepts the threshold of the neurons, making under-stimulated neurons more sensitive, and frequently stimulated neurons less sensitive. The average firing rate of a neuron would move toward some target rate. Combining the three synaptic plasticity rules we attempted to follow the Self-Organizing Recurrent Neural Network (SORN) approach from Lazar et al. (2009). Unfortunately, within our limited time-frame, we could not find appropriate parameters to make this approach work and will have to leave it as a suggestion for future research.

3.7 Inspecting Temporal Patterns

As explained before, for the actual predictions that are made by the LSM, the rate coding of each 10% of the input stream is fed into an SVR. However, the output is simply a single value so we resort to temporal voting to create the output. Instead of taking one or a few rate codings from the liquid in the LSM at the end of the sequence, we use temporal voting. It potentially yields better results to take outputs from multiple time points and let their averaged result be the final result. For the purposes of the current data, we take frames from

every 10% of the data, so after each 3 seconds of video. During training, the 10 selected states are trained with the same behavioral intention value, so that the classifier effectively learns that different rate codings from the liquid belong to an equivalent result. During testing, each frame can be given a vote and the average response of the votes will give the final value. In effect, votes are given for each temporal segment and their combined response may be more informative than only the last segment.

For any given data set, it may prove that the differences between examples (participants) are not as pronounced at all the particular points in time. Rather, there may be an optimal time to decide the output value for a particular example. We can demonstrate this by training a classifier on all the segments and measuring the classification error on separate segments on the test set. Instead of using the average of the different time frame measurements, we inspect each of the votes. In order to get meaningful results this inspection should be done over several different training/test set splits. We will plot the results of the average regression error over many random splits of training and test set (20 times sixfold cross-validation).

By inspecting the results at multiple points in time, we can achieve a better understanding of when important effects take place. If the error is low at a particular point in time for all participants, this is likely due to an important effect in the video. Combining feature selection with temporal inspection, we can answer very compelling questions. We can see if a particular feature or combination of features is important for particular segments in the video. As we are developing the methodology we will not hypothesize many possible effects of the interaction of the commercial and facial responses, but we will demonstrate the principle. To this end we plot the best features and their performance over time.

3.8 Summary

In this chapter, we have explained how we incorporate the temporal aspect of the data into our machine learning regression paradigm. A short background in recurrent neural networks was provided and we gave an in-depth account of Reservoir Computing. Most importantly, we detailed the inner workings of the Liquid State Machine and explained why it can achieve insightful computations. Nevertheless, it turned out that the behavior of an LSM is very

unpredictable and that we need to check many parameters and generate several liquids per parameter combination.

We expressed the importance of the network topology for connectivity of a liquid and thereby the computations on input. Finally, we explained how we intend to inspect temporal patterns using the LSM. The latter can be used to obtain insightful decision moments in the video (stimulus) that was played to the participants in our data set. In the next chapter, all results from these explorations are provided.

4 Results

Following the approach to explore machine learning in the field of behavioral marketing, we will state our results on our example data set in this chapter. We will first give a classical analysis of the type that is normally applied to behavioral marketing problems. Thereafter, we set a baseline performance with machine learning that is applied to increasingly sophisticated simplifications of the temporal dimension. The LSM and temporal analysis of the problem will then be detailed, while optimizations of parameters from several different topologies are given. Finally, the results of feature selection will be stated and explained.

4.1 Classical Analysis

Recall that in testing the first hypothesis we examined whether the following null hypothesis could be rejected:

H_0: The average happiness of the participants in the hindering condition was not significantly greater than in the control condition.

This translates into a one-tailed test to see if the mean of the averages in the hindering group was not significantly greater than the control. Unfortunately, it turns out that we could not reject the null hypothesis: t(93.79) = -1.5638, p-value = 0.061. It should be noted that we did not check for normality: inspection shows that the data is actually heavily skewed. A Mann-Whitney test (the non-parametric variation of the t-test) also showed no significant increase in happiness ($p > 0.5$). The much worse performance of the Mann-Whitney test demonstrates that the minor effect seen in the t-test is most likely due to the strong skewed

distribution while for the test it is assumed that the distribution is normal.

Surprisingly, Lewinski et al. (2016) did find an effect of the condition on the happiness of participants' emotions, albeit in a mediation model. They found no significant effect of the condition on the behavioral intentions directly, but did find it when the facial expressions of happiness were used as a mediating factor. The selection of participants was different in that study, as is the means of analysis, but we cannot confirm their conclusion with these more basic methods. Moreover, reapplying the above hypothesis test to the behavioral intention "attitude towards the ad" (AAD) did reject that there was no higher buying intention (W = 1425.5, p = 0.018). According to this analysis, the effect of the condition can be directly observed in the behavioral intention AAD, but not in the facial expressions of happiness of participants.

The lack of distinctive facial expressions does not mean that they cannot be used for prediction. The correlations in Table 4.1 were calculated to support our second hypothesis and from the educated guess we made for which features and transformations on these features should be important for behavioral intentions. Change indicates the difference between the first and second half of the video (AB refers to "attitude towards the brand" and BI to "buying intention").

	Behavioral Intentions		
Feature	AAD	AB	BI
Neutral Average	-0.039	-0.179^{2}	-0.119
Happy Average	0.282^{1}	0.177	0.182^{2}
Surprised Average	-0.067	0.094	-0.080
Valence Average	0.082	0.013	-0.013
Neutral Change	0.104	0.002	-0.047
Happy Change	0.314^{1}	0.280^{1}	0.248^{1}
Surprised Change	-0.131	-0.062	-0.082
Valence Change	0.360^{1}	0.283^{1}	0.187^{1}

Table 4.1: Spearman correlations between the best average and hardcoded features. [1] are significant with $p < 0.01$ and [2] at $p < 0.05$

In Figure 4.1 we offer two examples of the correlations plotted. Clearly, the strongest

correlations are between the features and the Attitude toward the Ad. Therefore, in the following we will focus our analysis on how well we can predict the AAD from the facial features.

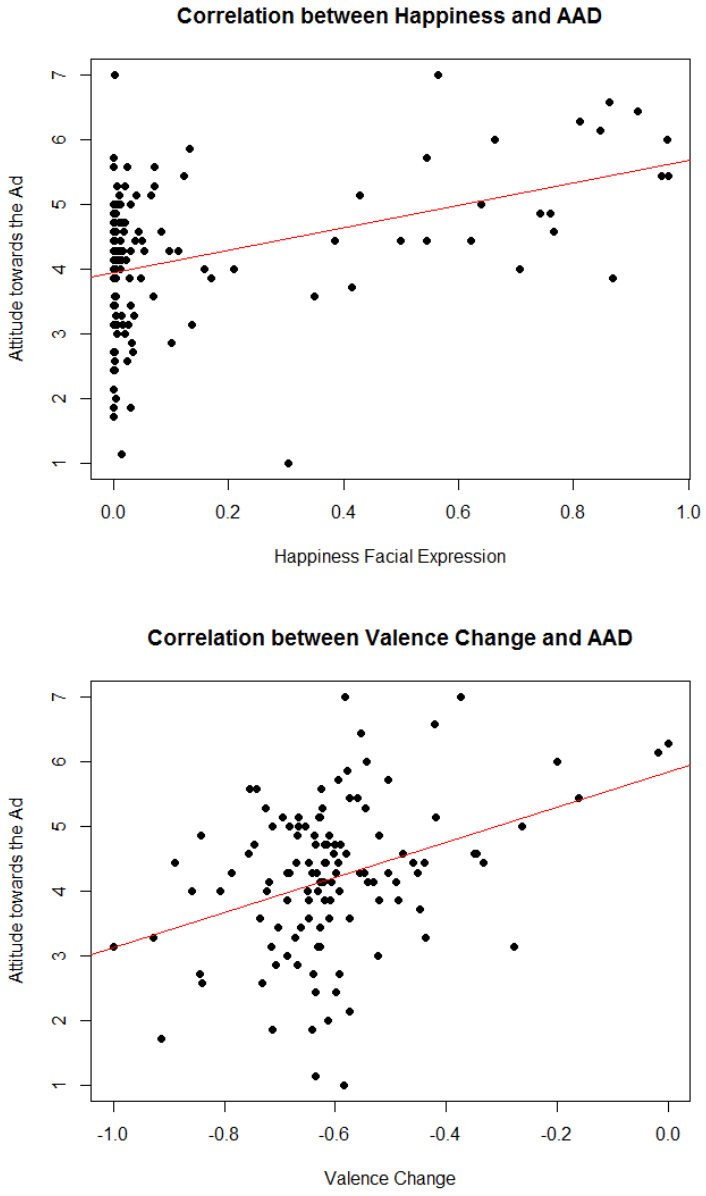


Figure 4.1: Correlation scores between emotions and behavioral intentions

4.2 Machine Learning Regression and Setting a Baseline

First, after extensive cross-validation on the parameters we could set with the neural network and SVR, we concluded that the SVR delivered the best performance, if by a narrow margin (no significant differences). We show an example of the validations procedure in Figure 4.2. This example was of a neural network with 5 hidden nodes and a learning rate of $\eta = 0.1$, for a regression of the hard-coded temporal features. As explained in the methods section, the weights of the hidden layer are saved for the lowest point of the validation set. For this particular example, that event took place at epoch 250 where the training set had an error of 0.968 and the validation set 1.094. The test set's error was actually lower than the validation set at 1.014. After more epochs the validation set error goes up, but the training set error gradually descended to an error of 0.660 at the last epoch we run. This clearly indicates that overfitting occurs and early stopping was needed to prevent this. Moreover, this result seems very good, but it is of only one devision of the sixfold cross-validation. The participants in the validation and test set may have been relatively easy to regress for the NN.

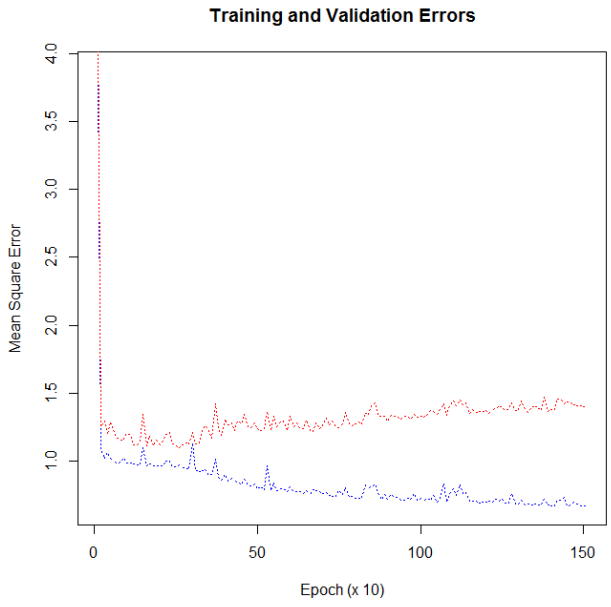


Figure 4.2: Training and validation set error for a neural network

Figure 4.3 contains an SVR parameter optimization of tolerance (of the termination criterion) and ϵ (allowed error), where $C = 1$ and $\gamma = 0.1$ were chosen from an optimization beforehand. This optimization was performed on the average of the emotions, and did include the full cross-validation, hence the ostensibly less impressive results. Good results were found for relatively high tolerance and epsilon, indicating that a very strict model most likely led to overfitting.

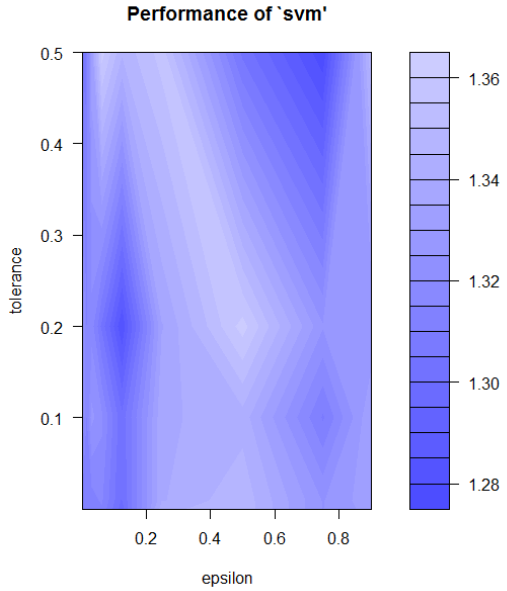


Figure 4.3: Training and validation set error for support vector regression

The optimum performance for the NN was found at 2 nodes, with learning rate $\eta = 0.05$. Nevertheless, SVR outperformed it with *tolerance* $= 0.2$, $C = 0.5$; $\gamma = 0.1$; $\epsilon = 0.15$ and radial basis as kernel. Other parameters were kept at their standard values. The difference in computational effort and thus calculation speed did offer a great advantage to the SVR. As the ANN implementation gave a good indication of what to expect in terms of results in the SVR, but lacked the speed, we concluded it would be wise to further use SVR. Moreover, when the liquid of the LSM is trained, speed is of great importance, as the liquid can be quite large and therefore it takes longer to calculate the model. For the LSM, the SVR parameters

C and ϵ were optimized for each liquid that is created. In other words, each model could have different parameters for the SVR (within the tested range).

In section 3.2 we introduced three baseline sets. After running these sets with the SVR for 20 times with sixfold cross-validation internally to obtain stable mean results, we conducted one-tailed paired t-tests. The MSE was significantly worse for the mean as predictor (M=1.38, SD = 0.57) compared to the average of the emotions (M = 1.26, SD = 0.02); $t(119) = 3.9915$, $p << 0.001$. In turn, the latter was significantly worse than the smart selection we made which included the hard-coded temporal features (M = 1.18, SD = 0.02); $t(119) = 29.540$, $p << 0.001$. The small p-values remain significant after Bonferroni correction. So our prediction about the ordering of the sets holds true. This was not a surprise because of the higher correlations with the behavioral intentions found in the hard-coded features.

In summary, we have demonstrated that it is possible to make predictions about the behavioral intentions of participants of whom we only inspect facial expression data. We have shown that prediction using the average over time per participant with ANN and SVR gives a significant improvement over using the mean over all participants as an estimator. In addition, hard-coding temporal elements and selecting the most discriminative features further improves regression. In the next section we will compare these results with the method that actually makes use of the temporal information of the facial expressions.

4.3 LSM

Now we turn to the LSM, the technique with which we aim to extract significant calculations from temporal patterns in the data. Initially, the results seemed not to be improving for the LSM, with an MSE of the mean of the temporal votes from 10 segments of around 1.30. However, when we inspect the errors over time, a clear pattern is unveiled, as shown in Figure 4.4. This figure reveals that if the votes were taken by themselves to predict the behavioral intentions, it is almost impossible to make sense of the first half of the video. This may actually reflect the liquid of an LSM takes some time to get into excited states that will separate the input better. Therefore, we trained SVR in the LSM only after half of the sequence had excited the liquid. As the second major remark, we notice that there is a point

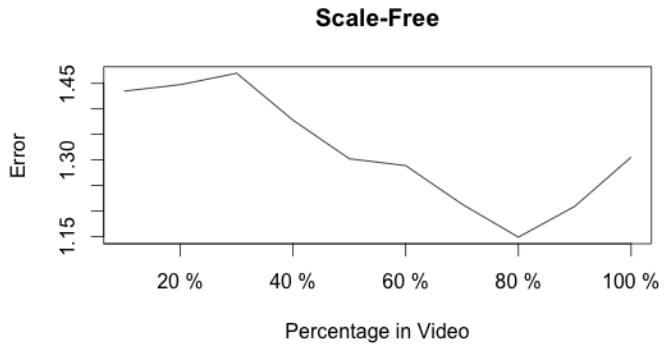


Figure 4.4: The error on AAD over the whole temporal segment

at 80% in the sequence where the vote only has a mean MSE of 1.149. We would like to find out if this pattern can still be observed if we only train SVR on the second half of the liquid's states (or rather the rate coding thereof).

After extensive optimization we found an LSM (Scale-Free) which achieved a mean MSE of 1.160, with at the 80% point in the video a mean MSE of just 1.097. The finding is produced in Figure 4.5. Indeed, the pattern is observed again, the lowest MSE is achieved after the second interval with the rate coding over 70% - 80% in the video. However, it should be noted that this was just one liquid that happened to perform well. If we take the average over several generated liquids, we can see a somewhat more balanced picture. We can observe this in terms of parameter optimizations per architecture. First we inspect the Waxman architecture in Figure 4.6

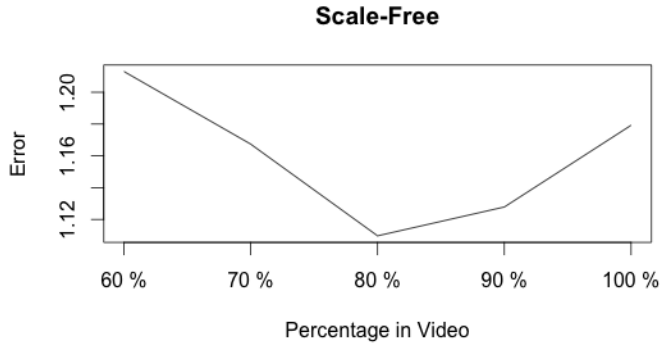
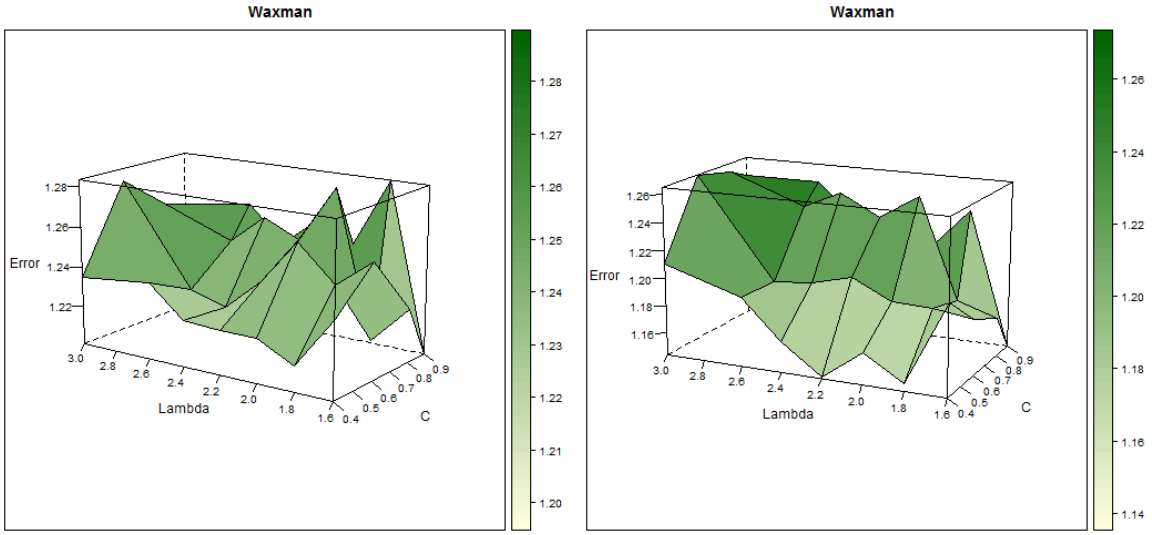


Figure 4.5: The error on AAD over the second half of temporal segment

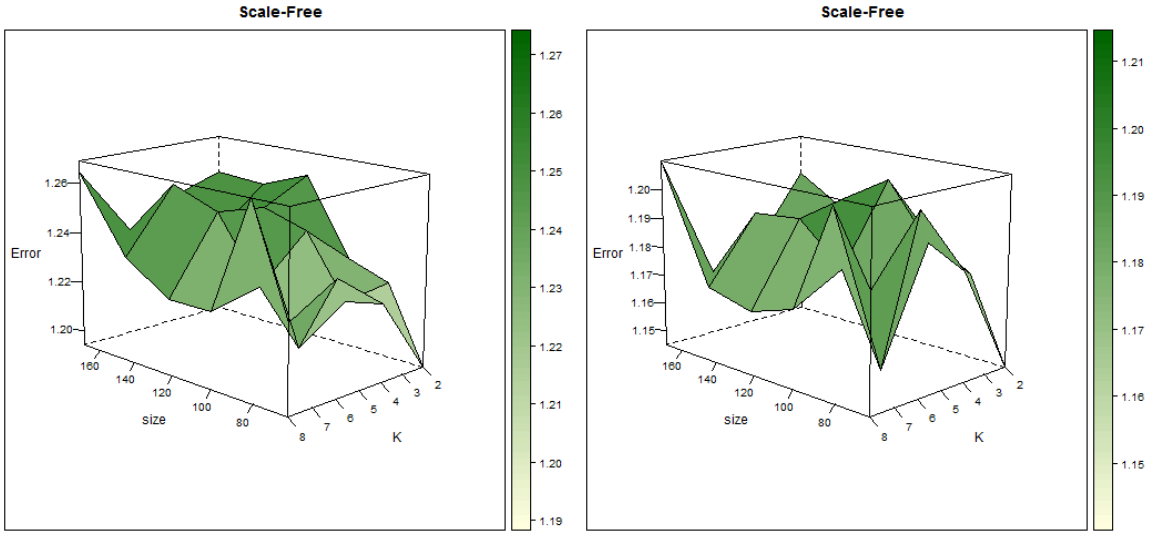


(a) Mean performance (b) Performance at 80%

Figure 4.6: Parameter optimization for Waxman topologies (averaged over multiple liquids)

We can see that the connection parameter λ should not be set too high, indicating that most likely the inner activity will start to dominate the input patterns, leading to worse results. Moreover, C appears to have an influence on the performance, where low C appears to be preferred, again indicating lower connectivity leads to better results. Nevertheless, we see that at $C = 0.9$ and $\lambda = 1.6$ the result is actually the best. The size of the liquid was set beforehand at $N = 144$, which during earlier optimization was the size after which no performance gain was noticed from increasing the size

Next, we tested whether the Scale-Free architecture parameter optimization actually gave some important improvement. The results can be seen in Figure 4.7. Normally, we would expect that larger liquids perform better on most data, but here we get the counter indication that the Scale-Free topology performs well at size $N = 64$. This may be an indication that the regression problem is very hard and there are a lot of poorly contributing features, which make the approximation more difficult in larger liquids. In addition, due to the small data



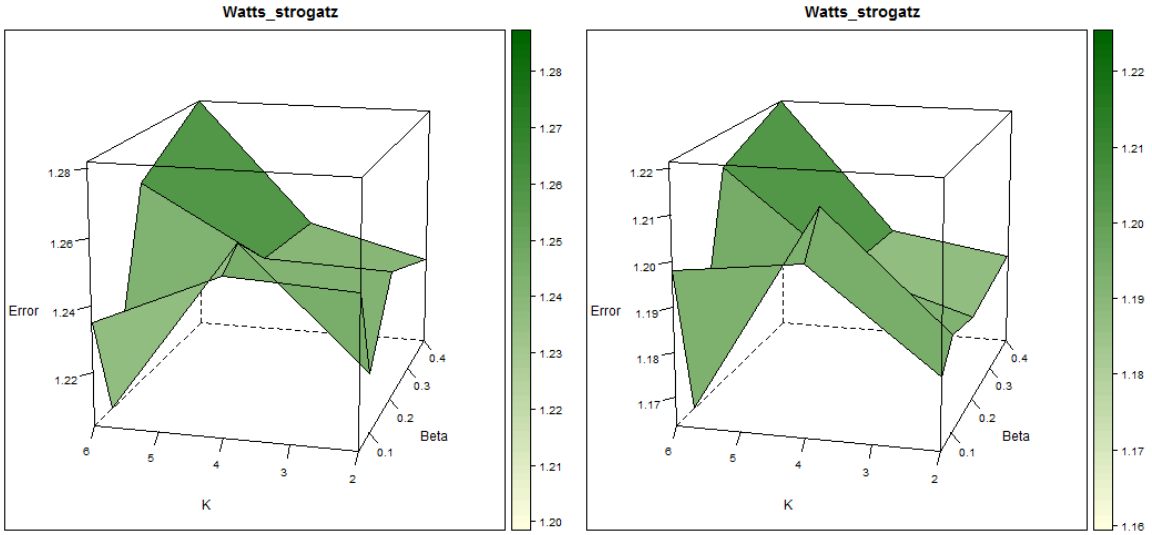
(a) Mean performance (b) Performance at 80%

Figure 4.7: Parameter optimization for Scale-Free topologies (averaged over multiple liquids)

set, lower complexity models may be preferred because they naturally counter overfitting to a certain extent.

Lastly, we search for optimal parameter setting in the Watts-Strogatz model (Figure 4.8). Again we see a preference for lower complexity of the network in the Watts-Strogatz performance. At $\beta = 0.1$ and $k = 6$, we basically have relatively many connections, but they remain connected mostly to their neighbors. It is unfortunate that we have a relatively small data set over which it seems hard to generalize.

For each of the liquids we kept track of the best performing liquids, and compared them to the average results of the optimization with the same parameter settings. As can be seen in the figures, the average results at these parameters were still among the best. The results are summarized in Table 4.2.



(a) Mean performance

(b) Performance at 80%

Figure 4.8: Parameter optimization for Waxman (averaged over multiple liquids)

	MSE			
Architecture	Best at 80%	Best Mean	Average at 80%	Average Mean
Scale-Free	1.097	1.160	1.145	1.194
Waxman	1.098	1.175	1.144	1.201
Watts-Strogatz	1.137	1.187	1.163	1.204

Table 4.2: Mean MSEs for the best parameters of each architecture of the liquid. Scale-Free at $N = 64$, $m = 2$, Waxman at $C = 0.9$, $\lambda = 1.6$, Watts-Strogatz at $\beta = 0.1$, $k = 6$. "Best Mean" indicates that multiple votes are taken from the entire length of the output (here only the second half of the sequence), whereas at "80%" means that only the vote from the liquid at 80% in the sequence is taken.

Finally, we should perform statistical analysis to establish whether the performance of an LSM is significantly better than that of the non-temporal SVR from the previous section. We test the best Scale-Free architecture over the second half of the liquid and at the 80% point. Earlier we concluded that the hard-coded feature set had the lowest MSE of the

non-temporal techniques (M = 1.18, SD = 0.02). Unfortunately, due to the design of our algorithms, we cannot conduct a paired t-test, we cannot directly compare the splits in data. From the unpaired t-test, we cannot conclude here that our best LSM (M = 1.16, SD = 0.36) performs significantly better. The standard deviation of the MSE from the LSM is too large to conclude anything. On the other hand, it is easy to show with a paired t-test that the minimum of that LSM at 80% (M = 1.10, SD = 0.37) has a significantly lower mean MSE than the average of the multiple votes; $t(119) = 4.685$, $p << 0.001$. At least we can conclude that through inspection of the liquid we could find a temporal segment that is significantly more indicative of the behavioral intention AAD than the whole sequence.

4.4 Feature Selection

The results from feature selection are very straightforward. As discussed in Chapter 2, we first eliminated the six features that contributed least to the performance of the LSM to predict behavioral intentions from facial expressions. The training and retraining was done with a Scale-Free LSM of size $N = 81$ and $m = 6$, which performed best in earlier results on all features. In order of elimination (from least contributing to relatively most contributing) they are:

- Scared
- Angry
- Valence
- Arousal
- Contempt
- Neutral

It can be deduced that the remaining features are: Happy, Sad, Surprised and Disgusted. For these four remaining features, we tried all possible combinations to gain insight in their interaction. Note that we plot the average of multiple votes. We see a very clear picture:

Feature Combination	Result
Happy	1.151
Happy + Sad + Disgust	1.152
Happy + Disgust	1.162
Happy + Sad + Surprise + Disgust	1.169
Happy + Surprise + Disgust	1.170
Happy + Sad	1.176
Happy + Sad + Surprise	1.188
Happy + Surprise	1.201
Surprise	1.234
Sad	1.253
Sad + Disgust	1.255
Disgust	1.268
Sad + Surprise + Disgust	1.272
Sad + Surprise	1.280
Surprise + Disgust	1.285

Table 4.3: MSEs of AAD predicted from LSMs trained on Combinations of the best four features

It becomes immediately evident that Happiness is the most significantly influential feature. By itself it performed the best, and all the combinations of happy and other features are ranked above other features by themselves or combined. A statistical analysis of all of these features would be somewhat to complex for the purposes of this thesis. A straightforward indication from these ranking results complement the earlier correlations we found between (transformations of) Happiness and the behavioral intentions.

When we inspect the temporal patterns for separate features, we indeed see the same pattern as we saw on all features together, although slightly less pronounced:

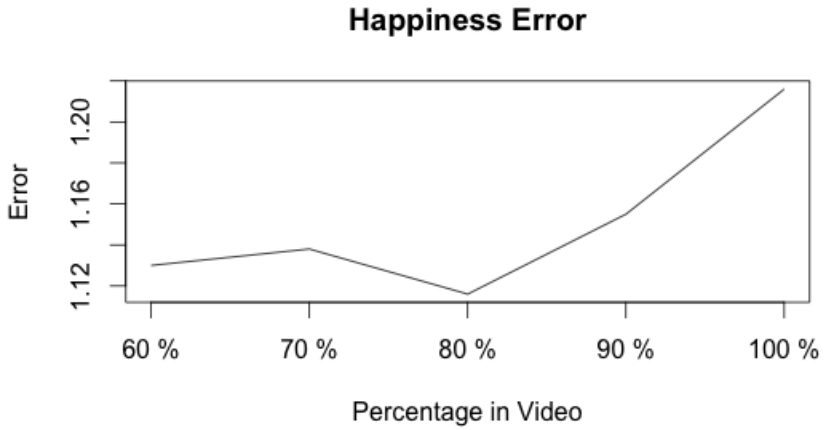


Figure 4.9: The error for the feature happiness

The average is stated in the table and here we can add that the lowest value is 1.107, at the 80^{th} percentile. Two other interesting remarks can be made when inspecting the temporal patterns in these features. Although the feature Sad has worse performance, it does have a more stable pattern over the last stages as the error does not rise toward the end. Finally, if we inspect the combination of all the features, we see that the best performance is actually in the 70^{th} percentile.

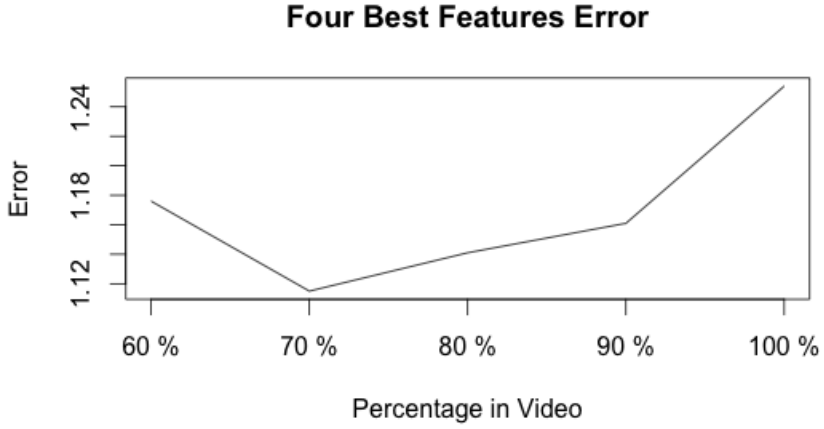


Figure 4.10: The error for the four best features combined: Happy, Surprise, Sad, Disgust

5 Conclusion

We have attempted to synchronize traditional methods from behavioral marketing research with machine learning. The benefit of this approach is obtaining data-driven, instead of theory-driven results. We constructed a prediction paradigm, in which we applied methods to uncover *which* features were important for prediction, as well as *where* they played an important part. It turned out that Happiness was the main driving feature, which agrees with the theory rather nicely. Moreover, it turned out that the time frame 70% - 80% was the most indicative segment in the video. This may indicate that the reaction, in terms of the facial expression Happiness, to a clever twist in the video, is ultimately the best predictor of "attitude toward the advertisement." This data-driven result agrees with the theory-driven results from earlier papers. Perhaps it is unfortunate that we do not find some unexpected effect, but at least this gives strong backing to the theory while providing a new way to inspect *where* in a time series an effect occurs.

So we have indeed accomplished benefits in behavioral marketing analysis from using machine learning. We can now formulate concrete answers to the research questions we have asked at the beginning of the thesis:

1. Can we use a Liquid State Machine to interpret behavioral data?
   We have made significant improvements in the analysis of behavioral data by applying the LSM to its temporal aspect. The machine learning paradigm of predicting the results of a test set has ultimately proven to be very useful in providing the setting for feature selection and inspection of the votes from an LSM.

2. Does the analysis of behavioral data with an LSM benefit from feature selection and from tracking the LSM's accuracy through time?

For our data set, it turned out that feature selection confirmed important hypotheses from theory. In addition, this data-driven approach may indicate directions for further research if other (unexpected) important features are found. The accuracy on different segments of the input data may uncover important patterns in the data, as was the case for our data set.

3. Can we minimize the parameter search for finding a well performing liquid by choosing a particular topology or applying a learning rule to update the connections in the liquid? On the basis of our data-set we would like to recommend the Scale-Free topology, which achieved good results, even for smaller sizes. However, it should be noted that we could not circumvent extensive testing and generating multiple liquids for each parameter setting in order to find a good liquid. In other words, we cannot give a general recommendation that would suffice for any data set. Although synaptic plasticity rules could potentially optimize a liquid after it has been generated, they add much computational time.

We cannot claim any breakthrough method for the parameter optimization, although we enjoy the results of the Scale-Free network. The third research question could not be answered positively, but we are very happy that we can affirm the first two research questions. We have successfully enhanced the insight in data from behavioral marketing experiments which have temporal aspects. It is our conviction that this success can be replicated in other (behavioral sciences) fields which deal with similar data and hope that in the future the methods we describe will be more commonly applied.

5.1 Suggestions for Future Research

In this thesis, we focussed mainly on the inner workings of the LSM, combined with some ideas about topologies. Due to the optimization time, fairly obvious extensions such as ensembles of LSMs or hierarchical LSMs were not considered. These may clearly improve the performance of the LSM. More importantly, some ideas that were considered could not be presented throughout this thesis. Here we would like to suggest some ideas which we could

not fully implement or test due to time constraints. There were three main ideas: a different activation model for the liquid, an entirely different LSM architecture and the application of unsupervised learning in the liquid.

The spiking neural network could have a different activation function. By changing the activation model, the number of parameters can be reduced. The Resonate and Fire (RaF) model (Izhikevich, 2001) is an example of this, as the leaking rate is implicit in the resonance of a neuron. In the RaF model, a neuron does not simply integrate over all its incoming connections and fires if it reaches a threshold, but instead has its own wave-like dynamic in response to an input. This behavior is observed in some neurons in the brain. In addition, the RaF model may induce more separation in the liquid, which should lead to better performance (Grzyb et al., 2009). However, on our data, initial experimentation did not give any better separation, though it may be possible that the performance could be better for other data sets, or if we try more parameters in the liquid optimization procedure.

Another idea that unfortunately did not reach fruition was that of creating a convolutional LSM. Just as visual features are extracted in deep learning architectures by calculating weights through a convolution of the picture with a mask, the same idea could be applied to the states of an LSM. 3-dimensional convolution could be used by viewing the LSM activation states as a video. However, whereas the pictures and videos have information from the physical world, in which objects are connected just by their physical properties, there is no guarantee that the same should be true for 2-dimensional activation patterns in brain-like structures. In other words, there is no sound theoretical justification for such an approach. After brief experimentation and disappointing results, we halted the development of this idea and would not recommend it for further research.

Lastly, we already discussed unsupervised learning rules in Chapter 3.6. We would recommend combining the three well-known synaptic plasticity rules into the Self-Organizing Recurrent Neural Network (SORN) approach from Lazar et al. (2009). The optimization of the liquid may make it more suitable to process the type of input data.

Bibliography

Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–1183.

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, Cambridge, MA, 2nd edition.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Bullmore, E. and Sporns, O. (2009). Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Cootes, T. F., Taylor, C. J., et al. (2004). Statistical models of appearance for computer vision. Technical report, University of Manchester.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Den Uyl, M. and Van Kuilenburg, H. (2005). The FaceReader: Online facial expression recognition. In *Proceedings of Measuring Behavior*, volume 30.

Ekman, P. (1972). Universal and cultural differences in facial expression of emotion. In *Nebraska symposium on motivation*, volume 19, pages 207–284, Lincoln. University of Nebraska Press.

Ekman, P. and Keltner, D. (1970). Universal facial expressions of emotion. *California Mental Health Research Digest*, 8(4):151–158.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Erdős, P. and Rényi, A. (1960). On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61.

Gómez, V., Kaltenbrunner, A., López, V., and Kappen, H. J. (2008). Self-organization using synaptic plasticity. *arXiv preprint arXiv:0808.3129*.

Graves, A., Mohamed, A.-R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE.

Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552.

Grzyb, B. J., Chinellato, E., Wojcik, G. M., and Kaminski, W. A. (2009). Which model to use for the liquid state machine? In *International Joint Conference on Neural Networks, IJCNN 2009.*, pages 1018–1024. IEEE.

Hager, J. C., Ekman, P., and Friesen, W. V. (2002). Facial action coding system. *Salt Lake City, UT: A Human Face*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A field guide to dynamical recurrent neural networks.* IEEE Press.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Izhikevich, E. M. (2001). Resonate-and-fire neurons. *Neural networks*, 14(6):883–894.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34.

Khushaba, R. N., Wise, C., Kodagoda, S., Louviere, J., Kahn, B. E., and Townsend, C. (2013). Consumer neuroscience: Assessing the brain response to marketing stimuli using electroencephalogram (EEG) and eye tracking. *Expert Systems with Applications*, 40(9):3803–3812.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lazar, A., Pipa, G., and Triesch, J. (2009). Sorn: a self-organizing recurrent neural network. *Frontiers in computational neuroscience*, 3.

Legenstein, R. and Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334.

Lewinski, P., den Uyl, T. M., and Butler, C. (2014a). Automated facial coding: Validation of basic emotions and FACS AUs in FaceReader. *Journal of Neuroscience, Psychology, and Economics*, 7(4):227.

Lewinski, P., Fransen, M. L., and Tan, E. S. (2014b). Predicting advertising effectiveness by facial expressions in response to amusing persuasive stimuli. *Journal of Neuroscience, Psychology, and Economics*, 7(1):1–18.

Lewinski, P., Tan, E. S., Fransen, M. L., Czarna, K., and Butler, C. (2016). Hindering facial mimicry in ad viewing: Effects on consumers' emotions, attitudes and purchase intentions. In *Advances in Advertising Research (Vol. VI)*, pages 281–288. Springer.

Lewis, D. and Phil, D. (2004). Market researchers make increasing use of brain imaging. *Nature Neuroscience*, 7(7):683.

Lukoševičius, M. and Jaeger, H. (2009). Survey: Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

Lundqvist, D., Flykt, A., and Öhman, A. (1998). The karolinska directed emotional faces (KDEF). *CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet*, pages 91–630.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.

Martens, J. (2010). Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742.

Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040.

McClure, S. M., Li, J., Tomlin, D., Cypert, K. S., Montague, L. M., and Montague, P. R. (2004). Neural correlates of behavioral preference for culturally familiar drinks. *Neuron*, 44(2):379–387.

Meyers-Levy, J. and Malaviya, P. (1999). Consumers' processing of persuasive advertisements: An integrative framework of persuasion theories. *The Journal of Marketing*, pages 45–60.

Milgram, S. (1967). The small world problem. *Psychology today*, 2(1):60–67.

Natschläger, T., Bertschinger, N., and Legenstein, R. (2005). At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. *Advances in neural information processing systems*, 17:145–152.

Noldus (2013). FaceReader: Tool for automatic analysis of facial expression (version 5.0.15).

Noldus (2015). FaceReader product page. http://www.noldus.com/human-behavior-research/products/facereader [Accessed: 20.04.2015].

Olson, J. C. and Mitchell, A. A. (2000). Are product attribute beliefs the only mediator of advertising effects on brand attitude? *Advertising & Society Review*, 1(1).

Pape, L., de Gruijl, J., and Wiering, M. (2008). Democratic liquid state machines for music recognition. In *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks*, pages 191–215. Springer.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Teixeira, T., Picard, R., and el Kaliouby, R. (2014). Why, When, and How Much to Entertain Consumers in Advertisements? A Web-Based Facial Tracking Field Study. *Marketing Science*, 33(6):809–827.

Teixeira, T., Wedel, M., and Pieters, R. (2012). Emotion-induced engagement in internet video advertisements. *Journal of Marketing Research*, 49(2):144–159.

Van Kuilenburg, H., Wiering, M., and Den Uyl, M. (2005). A model based method for automatic facial expression recognition. In *Machine Learning: ECML 2005*, pages 194–205. Springer.

Vapnik, V. (2000). *The nature of statistical learning theory*. Springer Science & Business Media.

Vapnik, V., Golowich, S. E., and Smola, A. (1997). Support vector method for function approximation, regression estimation, and signal processing. *Advances in neural information processing systems*, pages 281–287.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag, New York.

Vecchiato, G., Astolfi, L., De Vico Fallani, F., Toppi, J., Aloise, F., Bez, F., Wei, D., Kong, W., Dai, J., Cincotti, F., et al. (2011). On the use of EEG or MEG brain imaging tools in neuromarketing research. *Computational intelligence and neuroscience*, 2011:3.

VicarVision (2015). Product description. http://www.vicarvision.nl/facereader/productdescription/ [Accessed 20.04.2015].

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.

Waxman, B. M. (1988). Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University.

Williams, D., Rumelhart, G., Hinton, R., and Hinton, G. (1986). Learning representations by back-propagating errors. *Nature*, pages 323–533.