Language comprehension as a multiple label classification problem

R. Harald Baayen[1], Tino Sering[1], Cyrus Shaoul[2], Petar Milin[3]

[1] Eberhard Karls University of Tübingen, Germany
[2] Landmark College, USA
[3] University of Sheffield, UK

E-mail for correspondence: `harald.baayen@uni-tuebingen.de`

Abstract: The initial stage of language comprehension is a multi-label classification problem. Listeners or readers, presented with an utterance, need to discriminate between the intended words and the tens of thousands of other words they know. We propose to address this problem by pairing a network trained with the learning rule of Rescorla and Wagner (1972) with a second network trained independently with the learning rule of Widrow and Hoff (1960). The first network has to recover from sublexical input features the meanings encoded in the language signal, resulting in a vector of activations over all meanings. The second network takes this vector as input and further reduces uncertainty about the intended meanings. Classification performance for a lexicon with 52,000 entries is good. The model also correctly predicts several aspects of human language comprehension. By rejecting the traditional linguistic assumption that language is a (de)compositional system, and by instead espousing a discriminative approach (Ramscar, 2013), a more parsimonious yet highly effective functional characterization of the initial stage of language comprehension is obtained.

Keywords: multi-label classification, language comprehension, error-driven learning, Rescorla-Wagner, Widrow-Hoff

Table 1 presents 10 simple sentences. When reading these sentences, the letters and their combinations succeed in bringing to the fore a small number meanings while dismissing thousands of others as irrelevant. Sentences present the reader with a multi-label classification problem.
We address this problem as follows. First, we represent the orthographic input by means of letter trigrams. For the first sentence, these are `#Ma Mar ary ry# y#p #pa pas ass sse sed ed# d#a #aw awa way ay#` (the `#` symbol represents the space character). Letter trigrams provide a much

This paper was published as a part of the proceedings of the 32nd International Workshop on Statistical Modelling (IWSM), Johann Bernoulli Institute, Rijksuniversiteit Groningen, Netherlands, 3–7 July 2017. The copyright remains with the author(s). Permission to reproduce or extract any parts of this abstract should be requested from the author(s).

richer representation of the visual input than do orthographic words. For the data in Table 1, there are $n = 104$ distinct letter trigrams, to which we refer as cues.
The second column lists the lexical meanings (lexomes) that are the targets of classification. Lexomes are pointers to locations in a high-dimensional semantic vector space (defined below). Note that past-tense word forms such as *passed* (regular) and *ate* (irregular) are coupled with the lexomes PASS and EAT as well as with past tense (PAST). Likewise, the two word forms *apple* and *pie* are coupled with one lexome APPLEPIE, and the three expressions with the word forms *kicked the bucket*, *passed away*, and *died*, are all linked with the same lexome DIE.

TABLE 1. Sentences, lexomes in the message, and frequency of occurrence (F). The total number of learning events is $k = 771$.

	Sentence	Lexomes in the message	F
1	Mary passed away	MARY DIE PAST	40
2	Bill kicked the ball	BILL KICK PAST DEF BALL	100
3	John kicked the ball away	JOHN KICK PAST DEF BALL AWAY	120
4	Mary died	MARY DIE PAST	300
5	Mary bought clothes for the ball	MARY BUY PAST CLOTHES FOR DANCEPARTY	20
6	Ann bought a ball	ANN BUY PAST INDEF BALL	45
7	John filled the bucket	JOHN FILL PAST DEF BUCKET	100
8	John kicked the bucket	JOHN DIE PAST	10
9	Bill ate the apple pie	BILL EAT DEF APPLEPIE	3
10	Ann tasted an apple	ANN TASTE PAST INDEF APPLE	33

Is it possible to discriminate between the targeted lexomes given the letter trigrams in the sentences? We will show that considerable headway can be made by an error-driven incremental multi-label classifier that comprises two simple networks, each with only an input layer and an output layer. In what follows, we first provide a formal definition of the algorithm, and illustrate it for the sentences in Table 1. We then turn to a more realistic example in which lexomes targeted in around a million of utterances have to be discriminated from some 52,000 other lexomes.

1 An algorithm for multiple label classification

The problem of incremental learning of multi-label classification is defined by a sequence of events at which a set of features (henceforth cues) are present and generate predictions about classes (henceforth outcomes), only some of which are actually present in the learning event. The mismatch between predicted outcomes and the outcomes actually present in a learning event provides the error driving learning.

From a total of n distinct cues and m possible outcomes, only small subsets will be present in a given learning event. Let k denote the number of unique learning events (learning events may repeat, cf. *good morning* and *tickets please*). We index a specific learning event in the sequence $\mathbf{t}$ (of length $K \geq k$) of learning events by t. The classification problem is defined by $\mathbf{t}$, a sparse $n \times k$ cue matrix $\mathbf{C}$ which is 1 whenever a given cue is present in a specific event and zero otherwise, and a sparse $m \times k$ target matrix $\mathbf{T}$ that is 1 whenever an outcome is present and zero otherwise.
Classification proceeds in two steps, using two networks. The first network has cues as inputs and outcomes as outputs. It is defined by an $m \times n$ matrix $\mathbf{W}$ of connection weights from cues (columns) to outcomes (rows). Given $\mathbf{W}$, the predicted support (henceforth activation) for a specific outcome given the cues in the learning event is obtained by summation of the weights on the connections from these cues to that outcome. The $m \times k$ activation matrix $\mathbf{A}$ specifies these activations for all outcomes across all unique learning events:

$$\mathbf{A} = \mathbf{WC}.$$

The classification performance of this first network is assessed by checking whether the outcomes with the highest activations are those of the targeted lexomes.
As shown by Danks (2003), if over a sequence of learning events no further changes in the weight matrix take place other than the tiny increments and decrements that come with individual updates, i.e., when the weight matrix has entered a state of equilibrium, then, given the incremental learning rule of Rescorla and Wagner (1972) (see below), $\mathbf{W}$ can be estimated straight from conditional probabilities characterizing the input. Let $\mathbf{E}$ specify pairwise conditional probabilities of cues given cues,

$$\mathbf{E} = \begin{pmatrix} \Pr(c_0|c_0) & \Pr(c_1|c_0) & \dots & \Pr(c_n|c_0) \\ \Pr(c_0|c_1) & \Pr(c_1|c_1) & \dots & \Pr(c_n|c_1) \\ \dots & \dots & \dots & \dots \\ \Pr(c_0|c_n) & \Pr(c_1|c_n) & \dots & \Pr(c_n|c_n) \end{pmatrix},$$

and let $\mathbf{F}$ denote a matrix specifying conditional probabilities of outcomes given cues,

$$\mathbf{F} = \begin{pmatrix} \Pr(o_0|c_0) & \Pr(o_1|c_0) & \dots & \Pr(o_n|c_0) \\ \Pr(o_0|c_1) & \Pr(o_1|c_1) & \dots & \Pr(o_n|c_1) \\ \dots & \dots & \dots & \dots \\ \Pr(o_0|c_n) & \Pr(o_1|c_n) & \dots & \Pr(o_n|c_n) \end{pmatrix}.$$

Danks' equilibrium equations state that

$$\mathbf{F} = \mathbf{EW}^T,$$

which can be solved using the generalized inverse.

When a weight matrix is calculated in this way, the effect of the exact order of learning events is lost. Furthermore, a Danks weight matrix dampens the consequences of the frequencies of occurrence of cues and outcomes in the input space, while highlighting the contrasts that allow cues to discriminate between outcomes. Thus, the Danks weight matrix is useful when there is no information on the sequence of learning events (e.g., when only the frequency of learning events is available but not their order) and when interest is directed specifically to an idealised endstate of learning.
Preferably, the weight matrix $\mathbf{W}$ is estimated by repeated application of the learning rule of Rescorla & Wagner (1972) to the learning events $\mathbf{t}$. The update at learning event t,

$$\mathbf{W}^t = \mathbf{W}^{t-1} + \boldsymbol{\Delta}_{rw}$$

depends on the learning rate η (typically set at 0.001) regulating the magnitude of the changes to the weight matrix, on the predictions for the outcomes as gauged by the activations of these outcomes given the cues, and on whether the outcomes are actually present in the learning event. Specifically, let $\mathbf{c}$ denote the transpose of that column vector of $\mathbf{C}$ specifying which cues are present at the current learning event t, and let $\mathbf{o}$ denote the transpose of that column vector of $\mathbf{T}$ detailing which outcomes are present at t, and let $\mathbf{J}$ denote an $m \times n$ all-ones matrix. Let the (row) vector $\mathbf{a}_1$ to specify the activations of those outcomes that are present in the learning event while setting to zero the activations for all other outcomes:

$$\mathbf{a}_1 = (((\mathbf{J} \cdot \mathbf{o})^T \cdot \mathbf{c})^T \cdot \mathbf{W})\mathbf{i}.$$

Here, $\mathbf{i}$ is a row unit vector of length n. Note that $((\mathbf{J} \cdot \mathbf{o})^T \cdot \mathbf{c})^T$ is 1 for all cue-outcome combinations that are present in the learning event, and zero elsewhere. Next, let the (row) vector $\mathbf{a}_0$ represent the activations of those outcomes not present in the learning event, again given the cues in that learning event, and let it be zero for all other outcomes:

$$\mathbf{a}_0 = (((\mathbf{J} \cdot [\mathbf{1} - \mathbf{o}])^T \cdot \mathbf{c})^T \cdot \mathbf{W})\mathbf{i}.$$

$((\mathbf{J} \cdot [\mathbf{1} - \mathbf{o}])^T \cdot \mathbf{c})^T$ is 1 for all cue-outcome pairs where the cue is present but the outcome not, and zero elsewhere. The update to the weight matrix, $\boldsymbol{\Delta}_{rw}$, can now be defined as follows:

$$\boldsymbol{\Delta}_{rw} = \eta\{((\mathbf{J} \cdot \mathbf{o})^T \cdot \mathbf{c})^T \cdot (\mathbf{1} - \mathbf{a}_1) - ((\mathbf{J} \cdot [\mathbf{1} - \mathbf{o}])^T \cdot \mathbf{c})^T \cdot \mathbf{a}_0\}.$$

For cue-outcome pairs that are both in the learning event, the update of their weight is given by the difference from the maximal activation, 1 by definition. As the summed activations $\mathbf{a}_1$ tend to be less than 1, weights will be strengthened. For cue-outcome pairs where the cue is present but the outcome is not, the corresponding connection weight is decreased by the summed activations $\mathbf{a}_0$. Estimation of $\mathbf{W}$ using incremental updating

over the sequence of learning events is fast, first because only parts of the weight matrix require updating (efferent weights from cues not present in the learning event are left untouched), and also because the updates to individual outcomes are independent and hence allow for parallelization.
The activation matrix $\mathbf{A} = \mathbf{WC}$ specifies, for each unique learning event and for each outcome, the joint support provided by the cues in that learning event for that outcome.
Although class predictions based on $\mathbf{A}$ can do well for small constructed data sets, they lack precision for large real data sets. Prediction accuracy can be further improved by a second network that is given the task to predict the target $\mathbf{T}$ from the activation matrix $\mathbf{A}$:

$$\mathbf{T} = \mathbf{DA}.$$

The prediction matrix

$$\mathbf{P} = \mathbf{DA}$$

is the resulting approximation of $\mathbf{T}$. Although $\mathbf{D}$ can be calculated using the generalized inverse of $\mathbf{A}$, computation costs can be prohibitive for large numbers of learning events. It is therefore preferable to estimate $\mathbf{D}$ as follows:

$$\begin{aligned} \mathbf{T} &= \mathbf{DA} \\ \mathbf{TA}^T &= \mathbf{DAA}^T \\ \mathbf{Y} &= \mathbf{DX}, \end{aligned}$$

which leads to $\mathbf{D} = \mathbf{YX}^{-1}$. Since $\mathbf{X}$ is $m \times m$, and since generally $m \ll k$, computational costs are much lower when calculating $\mathbf{X}^+$ as compared to calculating $\mathbf{A}^+$.
The prediction matrix can also be estimated iteratively by means of the update rule of Widrow and Hoff (1960). This update rule, which specifies the update $\mathbf{\Delta}_{wh}$ to the $m \times m$ second weight matrix $\mathbf{D}$, is important, first, as it allows us to assess the consequences of how the order of learning events affects classification, and second, because for large numbers of training events (in the order of hundreds of millions), it is not feasible to actually calculate $\mathbf{A}$ (and $\mathbf{P}$).
Let $\mathbf{Z}$ denote an $m \times m$ matrix initialized with zeroes, let $\mathbf{a}$ denote the column vector of the activation matrix $\mathbf{A}$ giving the predicted activations for the current learning event, and let $\mathbf{o}$ denote the transpose of the corresponding column vector of the target matrix $\mathbf{T}$. The Widrow-Hoff update to $\mathbf{Z}$ is:

$$\mathbf{\Delta}_{wh} = \eta\{\mathbf{a}(\mathbf{o} - \mathbf{a}^T\mathbf{Z})\}.$$

We take the transpose to obtain $\mathbf{D} = \mathbf{Z}^T$.
The weights for the two networks ($m \times n$ for the Rescorla-Wagner network, and $m \times m$ for the Widrow-Hoff network) can be estimated in two ways.

One possibility is to first estimate **W** and then estimate **D**. Alternatively, one can update both networks in tandem for each successive learning event. In this case, it is not necessary to calculate **A**. Note that when estimating

$$\mathbf{P} = (\mathbf{WC})^{+}\mathbf{TWC}$$

we 'inject' error twice: once during the estimation of **W** and again during the estimation of **P**.

The equilibrium equations are implemented in the `ndl` package for `R` on CRAN. An efficient Python implementation for incremental learning of **W** is available at `github.com/quantling/pyndl`. An implementation of incremental learning for `R` is available (for `linux` only) upon request from the authors. Software for efficient updating of **D** by Widrow-Hoff is currently under development.

Returning to the example of Table 1, first consider classification performance when **W** and **D** are estimated independently, using incremental updating for the former, and the generalized inverse for the latter. In this case, for each of the 10 sentences, the lexomes in that sentence have the highest prediction values in **P**.

When the two networks are updated in tandem, with at each learning event first an update of **W** and then an update of **D**, accuracy varies with the (random) order in which the 771 learning events are made available to the model. For one such random order, the proper lexomes had the highest ranks in **A** for 9 out of 10 sentences. The one sentence with an error is *John kicked the bucket*, where DEF (the lexome for the definite article) intrudes with a higher activation before DIE, which is found at the next rank (4).

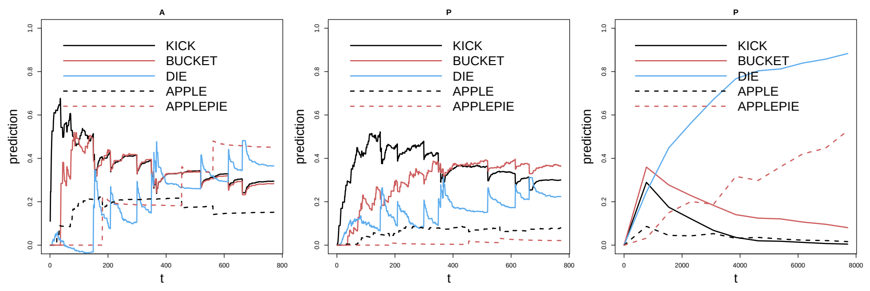

FIGURE 1. Prediction strengths for selected lexomes in the learning events of sentences 8 and 9 in Table 1, using incremented coupled Rescorla-Wagner and Widrow-Hoff. Left and center panels: frequencies as in the table; right panel: frequencies increased tenfold.

Figure 1 illustrates this incremental training regime. The left and center panels show the predictions based on **A** and **P** when training proceeds on a random order of 771 learning events, and the right panel when training proceeds on 7710 learning events. Solid lines represent key lexomes from sentence 8 in Table 1: KICK and BUCKET for the unintended literal reading and DIE for the intended idiomatic reading. Dashed lines represent the competitors APPLE and APPLEPIE in sentence 9. The spiky behavior in the left and center panels reflects the learning and unlearning that unfolds as outcomes competing for the same cues are encountered. Comparison of the left and center panels shows that the Rescorla-Wagner network learns much faster than the Widrow-Hoff network. By the end of the learning sequence, the former, but not the latter network succeeds in giving the intended lexomes higher prediction scores. The rightmost panel shows that with sufficient experience, the model learns that *kick the bucket* means DIE, and that an *apple pie* is not an apple but a particular kind of pie.

An important property of this approach to language comprehension is that the correct lexomes are selected without any worries about regular or irregular verbs, literal versus idiomatic expressions, finding boundaries between words, decomposing words into parts, or disambiguating homographs. Given the assumption that understanding drives the recalibration of weights, the rich information available in the combinatorics of sublexical cues and lexomes is sufficient for multiple label classification to be effective.

2 Multiple label classification with 52,000 classes

To clarify whether this approach scales up, we applied our algorithm to the TASA corpus (Zeno, 1995), a collection of texts comprising a total of 10,807,146 words representing 109,338 string types. Lemmatization was carried out with `TreeTagger` (`www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`), which distinguished 90,339 lemmata, of which 37,938 occurred once. To keep computations tractable, the model was trained on all words occurring at least twice and 351 hapax legomena that occurred in a precompiled list of words. Hapax legomena that were not included were replaced by the dummy word `HAPAX`, resulting in a total of 52,401 lexomes. Learning events were sentences in the TASA corpus. Sequences of more than 8 words were split at the next available occurrence of *and* or *or*. This resulted in a total of 992,752 learning events. The multi-label classification challenge is to predict the appropriate lexomes (out of 52,401) given the letter trigrams of the (possibly inflected) words in the learning events.

Using the `ndl2` package for R, **W** (52,401 lexomes $\times$ 11,724 letter trigrams) was estimated using all learning events. To keep computations tractable for the second network, two learning events were selected randomly from a precompiled list of 8866 targeted lexomes, resulting in a total of 17,455 learning events (in 276 cases there was overlap with two or more lexomes in the same event, and for one word, there was only 1 learning event available). The total number of outcomes in this subset of learning events was 19,020. With these restrictions, the matrices **A** (19,020 lexomes $\times$ 17,455 learning events), **D** (19,020 $\times$ 19,020 lexomes) and **P** (19,020 lexomes $\times$ 17,455 events) could be estimated straightforwardly.

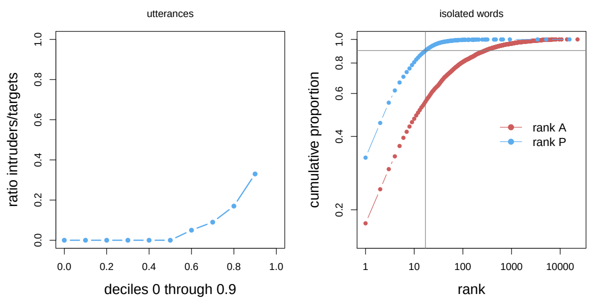

FIGURE 2. Left: Quantiles of the ratio of intruders (false positives) to targets (correct identifications), full utterances. Right: Rank and corresponding cumulative proportion based on **A** (red) and **P** (blue), isolated words.

The left panel of Figure 2 presents the ratio of intruders (lexomes with an activation exceeding that of the least activated target lexome) to the number of targeted lexomes. The median number of intruders is zero, at the 8th decile the ratio is 0.17, and at the 9th, it is 0.33. At the 10th decile, we find cases with vast numbers of intruders, leading to a maximal ratio of 1208.9. Examples of intruders are *down* for the sentence *The aleuts were housed in abandoned rundown gold mines or fish canneries*, and *field* and *success* for the sentence *He is an ecologist who studied succession in abandoned cornfields*.
We also tested identification performance when target lexomes were presented in isolation. The right panel of Figure 2 plots in blue cumulative proportion (out of a total of 7179) against rank based on **P**: 34% of lexomes had the highest prediction value, 88% of the targeted lexomes had at most a rank of 16 (indicating 15 intruders with higher activations). As show by the red curve, performance based on **A** instead of **P** is substantially worse. Human lexical decision performance, as gauged using the British Lexicon Project (BLP, Keuleers et al. 2012) was for the present data at 90% correct. As the lexical decision task does not require actual identification, but only sufficient evidence for lexicality, it appears that human subjects tolerate around 16 intruders.
As shown in Figure 3, the model also predicts power-transformed lexical decision response times ($t' = -1000t^{-1}$). For all but the first decile, log activation $a_i = \mathbf{W}\mathbf{c}_i$ (with $\mathbf{c}$ the vector specifying the present and absent cues in the input, and i indexing a specific lexome) shows a nearly linear effect with negative slope. Log rank prediction (the log rank of $p_i = \mathbf{D}\mathbf{W}\mathbf{c}_i$) has a smaller effect that is again negative and nearly linear, but now for the first nine deciles. The 90% decile of the rank is at 18, which is close to the cut-off at rank 17 for lexicality decisions in the right panel of Figure 2. Apparently, the same range of ranks influences both decisions and reaction times.

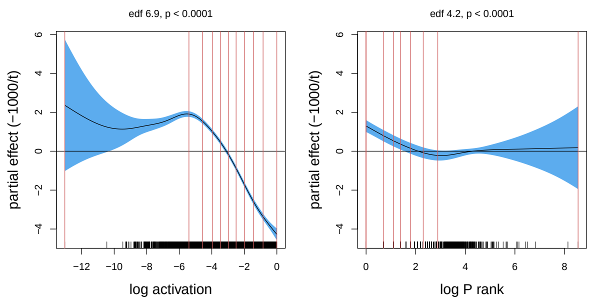

FIGURE 3. Partial effects in a GAM fitted to power-transformed ($-1000t-1$) reaction times. Left: log activation; Right: log prediction rank. Vertical lines denote deciles. The 90% decile of log prediction rank is at rank 18 (red lines indicate deciles). Regression analyses were carried out with GAMs (Wood, 2006).

$\mathbf{P}^T$ defines a semantic vector space (cf. Landauer & Dumais, 1997), and lexomes are indices or pointers for locations in this space. By way of illustration of the semantic nature of $\mathbf{P}^T$, the left panel of Figure 4 presents partial effects for human semantic similarity ratings for word pairs (Bruni et al., 2014) as predicted from correlations of the corresponding column vectors of $\mathbf{P}^T$ (left). For 90% of the data points, a nearly linear relation is observed. Clearly, extreme values are unreliable as predictors. Similarity in $\mathbf{P}^T$-space, i.e., similar prediction values across events and thus greater similarity of experiences communicated, correctly predicts greater perceived semantic similarity.

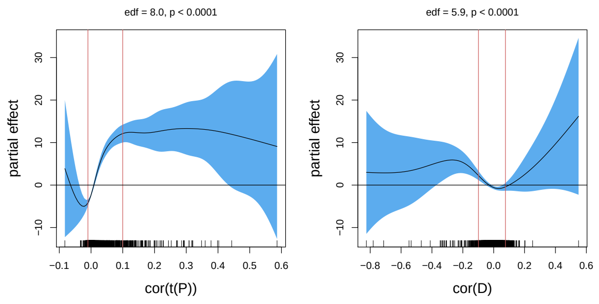

FIGURE 4. Partial effects of the correlations of row vectors of $\mathbf{P}$ (left) and column vectors of $\mathbf{D}$ as predictors of human similarity ratings for 2,369 word pairs. Red vertical lines indicate 5% and 95% percentiles. Regression analyses were carried out with GAMs (Wood, 2006).

The column vectors of $\mathbf{D}$ also define a lexomic space, but similarities in this space turn out to be positively correlated with the Levenshtein distance between the orthographic forms of the two words. As shown in the right panel of Figure 4, the more different two word forms are, the lower their perceived semantic similarity.

3 Concluding remarks

Multi-label classification is a hard problem, not only for statistics, but also for humans. For instance, in auditory word recognition, isolated words taken from conversational speech have recognition rates between 20% and 40% (Arnold et al., 2017). In the visual lexical decision task, undergraduate students perform near chance on the lower-frequency words (Baayen et al., 2017). From this perspective, the model's performance, with training on a mere 10 million words, is too good to be true. This is, of course, due to the model being given perfect feedback, whereas human learning tends to proceed under uncertainty and lack of full understanding.

Given that the model presents a simplified perspective on the first stage of comprehension — understanding the words — several of its features are remarkable. First, the traditional linguistic assumption that language is a (de)compositional system is replaced by a perspective in which the language signal is a code that discriminates between possible messages (Ramscar 2013, Shannon, 1956).

Second, the model is parsimonious with only one free parameter, the learning rate η. And although $\mathbf{W}$ and $\mathbf{D}$ can be very large, most of the weights are close to zero. E.g., for $\mathbf{W}$, only 5,885 weights exceed 0.1 (0.00058% of the total number of weights), and only 195 weights are greater than 0.5. Arnold et al. (2017) show for auditory comprehension that $\mathbf{W}$ can be pruned down to a fraction of the original weights without noticeable loss of accuracy.

Third, the classifier implements a three-layer network that differs from backpropagation networks in that there is direct error injection twice, once for $\mathbf{W}$ using the Rescorla-Wagner equations, and once for $\mathbf{D}$, using Widrow-Hoff (or the generalized inverse). Importantly, the power of the first network should not be underestimated. Although ever since the criticism of the perceptron by Minsky & Papert (1972), two-layer networks have been regarded as far too restricted for any classification tasks requiring more than the simplest linear separation, it turns out that actually, with an appropriate choice of cues, Rescorla-Wagner networks can solve much more interesting problems. Figure 5 illustrates this for a simple example with two classes (represented by gray and red points) that in $R \times R$ are not linearly separable (left panel). When the data are re-represented by identifiers for rows and columns (right panel), a Rescorla-Wagner network correctly predicts the highest activations for around 210 of the 260 elements of the red class (see Baayen and Hendrix, 2017, for detailed comparison with other machine learning classifiers, and also Ghirlanda, 2005).

Fourth, more sophisticated features than letter trigrams can be used as cues, such as the frequency band summary features used by Arnold et

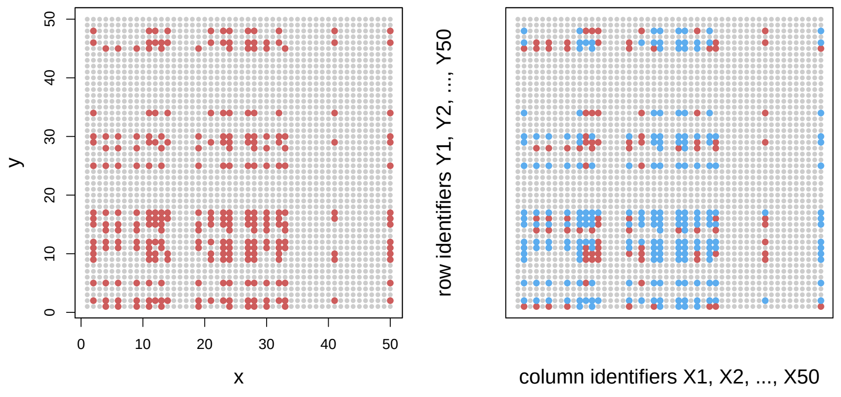

FIGURE 5. A non-linearly separable classification problem with a majority class in gray (2240) and a minority class in red (260). Left: data points in a Cartesian grid ($x = 1, 2, \ldots, 50; y = 1, 2, \ldots, 50$). Right: rerepresentation with row and column identifiers as cues for a Rescorla-Wagner network: hits in blue, misses and false alarms in red.

al. (2017) for modeling auditory word recognition, and for reading the histogram of oriented gradients feature descriptor proposed by Dalal and Triggs (2005).
Finally, the model is transparent to interpretation. **W** specifies the support provided by sublexical features for lexomes. **D** transforms activation vectors that are still strongly influenced by form similarity into vectors closer to the targeted lexomes, which in turn results in a semantic vector space, $\mathbf{P}^T$.

References

Arnold, D., Tomaschek, F., Sering, K., Lopez, F., and Baayen, R.H. (2017). Words from spontaneous conversational speech can be recognized with human-like accuracy by an error-driven learning algorithm that discriminates between meanings straight from smart acoustic features, bypassing the phoneme as recognition unit. *PLOS-ONE*, **12**, e0174623.

Baayen, R. H., and Hendrix, P. (2017). Two-layer networks, non-linear separation, and human learning. In Wieling, M., Kroon, M., van Noord, G., and Bouma, G. (Eds.) From Semantics to Dialectometry. Festschrift in honor of John Nerbonne. London, College Publications, 13–22.

Baayen, R. H., Tomaschek, F., Gahl, S., and Ramscar, M. (2017). The Ecclesiastes principle in language change. In Hundt, M., Mollin, S., and

Pfenninger, S. (Eds.) *The changing English language: Psycholinguistic perspectives.* Cambridge, Cambridge University Press.

Bruni, E. and Tran, N.K. and Baroni, M. (2014). Multimodal distributional semantics, *Journal of Artificial Intelligence Research*, **49**, 1–47.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR'05*, volume 1, 886–893.

Danks, D. (2003). Equilibria of the RescorlaWagner model. *Journal of Mathematical Psychology*, **47**, 109–121.

Ghirlanda, S. (2005). Retrospective revaluation as simple associative learning. *Journal of Experimental Psychology: Animal Behavior Processes*, **31**, 107–111.

Keuleers, E. and Lacey, P. and Rastle, K. and Brysbaert, M. (2012). The British Lexicon Project: Lexical decision data for 28,730 monosyllabic and disyllabic English words, *Behavior Research Methods*, **44**, 287–304.

Landauer, T.K. and Dumais, S.T. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. *Psychological Review*, **104**, 211–240.

Minsky, M. and Papert, S. (1969). *Perceptrons: An introduction to computational geometry.* Cambridge, MA: The MIT Press.

Ramscar, M. (2013). Suffixing, prefixing, and the functional order of regularities in meaningful strings. *Psihologija*, **46**, 377–396.

Rescorla, R. A. and Wagner, A. R. (1972). A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In: A. H. Black and Prokasy, W. F. (Eds.), *Classical conditioning II: Current research and theory.* New York: Appleton Century Crofts.

Shannon, C. E. (1956). The bandwagon, *IRE Transactions on Information Theory*, **2**, 3.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. *1960 WESCON Convention Record* Part IV, 96–104.

Wood, S. (2016). *Generalized additive models.* New York: Chapman & Hall.

Zeno, S.M. and Ivens, S.H. and Millard, R.T. and Duvvuri, R. (1995). *The educator's word frequency guide.* New York: Touchstone Applied Science.