Perspect Behav Sci (2018) 41:121–154
DOI 10.1007/s40614-017-0125-6

The Emergence of Stimulus Relations: Human and Computer Learning

Chris Ninness[1] · **Sharon K. Ninness**[2] · **Marilyn Rumph**[1] · **David Lawson**[3]

Published online: 13 November 2017

Abstract Traditionally, investigations in the area of stimulus equivalence have employed humans as experimental participants. Recently, however, artificial neural network models (often referred to as connectionist models [CMs]) have been developed to simulate performances seen among human participants when training various types of stimulus relations. Two types of neural network models have shown particular promise in recent years. RELNET has demonstrated its capacity to approximate human acquisition of stimulus relations using simulated matching-to-sample (MTS) procedures (e.g., Lyddy & Barnes-Holmes *Journal of Speech and Language Pathology and Applied Behavior Analysis*, *2*, 14–24, 2007). Other newly developed connectionist algorithms train stimulus relations by way of compound stimuli (e.g., Tovar & Chavez *The Psychological Record*, *62*, 747–762, 2012; Vernucio & Debert *The Psychological Record*, *66*, 439–449, 2016). What makes all of these CMs interesting to many behavioral researchers is their apparent ability to simulate the acquisition of diversified stimulus relations as an analogue to human learning; that is, neural networks learn over a series of training epochs such that these models become capable of deriving novel or untrained stimulus relations. With the goal of explaining these quickly evolving approaches to practical and experimental endeavors in behavior analysis, we offer an overview of existing CMs as they apply to behavior–analytic theory and practice. We provide a brief overview of derived stimulus relations as applied to human academic remediation, and we argue that human and simulated human investigations have

Electronic supplementary material The online version of this article (doi:10.1007/s40614-017-0125-6) contains supplementary material, which is available to authorized users.

✉ Chris Ninness
cninness@suddenlink.net

[1] Behavioral Software Systems, 2207 Pinecrest Dr, Nacogdoches, TX 75965, USA

[2] Texas A&M University—Commerce, Commerce, TX, USA

[3] Sam Houston State University, Huntsville, TX, USA

symbiotic experimental potential. Additionally, we provide a working example of a neural network referred to as emergent virtual analytics (EVA). This model demonstrates a process by which artificial neural networks can be employed by behavior–analytic researchers to understand, simulate, and predict derived stimulus relations made by human participants.

Keywords Contextual control · Stimulus equivalence · Connectionist model · Epochs · Momentum

In a chapter designed as an introduction to the basic principles and procedures revolving around stimulus equivalence, Critchfield and Fienup (2008) explored the historical and theoretical foundations of stimulus equivalence theory, discussed several studies in the general area of stimulus relations, and speculated on the extent to which related disciplines might contribute to the quickly growing area of derived stimulus relations. Within this chapter, the authors considered the extent to which artificial neural networks might be useful in furthering our understanding of derived stimulus relations and forwarded the notion that "neural networks are the inspiration for computer programs that can simulate complex psychological abilities. The software building blocks of these programs are intended to mimic interconnected networks of neurons that fire together under certain circumstances" (pp. 8–9). Although Critchfield and Fienup discussed a variety of related topics in their chapter, we will give special attention to stimulus relations as applied to human and simulated human learning and the convergence of these two seemingly dissimilar areas.

Equivalence Preparations via Matching-to-Sample

Stimulus equivalence is a behavioral phenomenon that is usually confined to humans with some degree of verbal proficiency and can be demonstrated in a diverse array of experimental preparations. Frequently, equivalence investigations are conducted by way of matching-to-sample (MTS) procedures that expose participants to a series of conditional discriminations using arbitrary stimuli (e.g., nonsense symbols or abstract designs). In more didactic academically focused investigations, selection of experimental stimuli may be based on the participants' lack of familiarity with particular types of symbols, concepts, or other academic subject matter.

Irrespective of the types of stimuli used within a given experiment, participants are taught to select a sample stimulus B in the presence of A (A → B). In the same way, C is trained as the correct response in the presence of B (B → C). Subsequent to such a training history, participants demonstrate an increased probability of selecting A from an array of comparison stimuli when B is presented (symmetry: A ⇆ B and B ⇆ A), selecting B from an array of comparisons when C is presented as a sample (symmetry: B ⇆ C and C ⇆ B), selecting A when C is displayed (transitivity: A ⇆ C), and selecting C when A is displayed (equivalence: C ⇆ A). Prior to the early research conducted by Sidman and colleagues (e.g., Sidman & Cresson, 1973; Sidman & Tailby, 1982), the emergence of the aforementioned derived stimulus relations was not anticipated during experimental preparations focusing on conditional discriminations.

Basic and Advanced Academic Remediation

In the decades following the seminal investigations by Sidman and colleagues (Sidman & Cresson, 1973; Sidman & Tailby, 1982), it became increasingly apparent that equivalence-based techniques had much to offer as a new instructional strategy for a wide range of academic and preacademic applications. For example, equivalence studies employed MTS procedures to teach reading skills (e.g., Connell & Witt, 2004), skills required for letter word recognition (e.g., Stromer, Mackay, & Stoddard, 1992), reading and spelling skills (e.g., De Rose, De Souza, & Hanna, 1996), and geographical locations (LeBlanc, Miguel, Cummings, Goldsmith, & Carr, 2003).

Aiming to develop more efficient instructional technologies, recent studies have used interactive equivalence-based protocols to convey challenging concepts to university students. For instance, equivalence-based strategies have been used to teach brain–behavior relations (Fienup, Covey, & Critchfield, 2010). Similar investigations with an emphasis on merging stimulus classes have been used to efficiently convey important concepts in the area of inferential statistics and hypothesis decision making (e.g., Critchfield & Fienup, 2010; Fienup & Critchfield, 2010; Fienup, Critchfield, & Covey, 2009). Fienup and Critchfield (2011) developed a more comprehensive investigation by blending many of their equivalence-based instructional tactics. Similar investigations have highlighted the potential for teaching university undergraduate students. For example, Walker, Rehfeldt, and Ninness (2010) taught stimulus relations focusing on disability names, definitions, etiologies, and commonly employed interventions. In a broadly similar vein, Lovett, Rehfeldt, Garcia, and Dunning (2011) employed an equivalence-based strategy to teach single-subject experimental designs to university students. Clearly, humans are able to learn more efficiently by way of equivalence-based instruction. Might the same be true for computers? Is it possible that we might use neural networks to vet the potential value of various equivalence-based interventions?

The Rise of Neural Networks

Following the work of McClelland and Rumelhart (1986), the development of artificial neural network software advanced and expanded exponentially. During the last 30 years, neural network technologies have become increasingly pervasive throughout the scientific and academic world. To list but a small fraction of the available examples, computer learning has been applied successfully to the understanding and prediction of environmental changes (e.g., Allamehzadeh & Mokhtari, 2003), the prediction of financial crises (e.g., Arciniegas, Daniel, & Embrechts, 2001; Erdal & Ekinci, 2013), and the identification and prediction of medical and physiological conditions (e.g., Abbass, 2002; Huang et al., 2013; Ninness et al., 2012; Wolberg, 1992; You & Rumbe, 2010). Precise forecasts have been demonstrated when attempting to predict severe weather patterns (e.g., Knutti, Stocker, Joos, & Plattner, 2003; Maqsood, Khan, & Abraham, 2004), predict the demand for electricity during severe weather conditions (e.g., Khan & Ondrusek, 2000; Oğcu, Demirel, & Zaim, 2012), identify probable terrorist locations (e.g., Guo, Liao, & Morgan, 2007), and classify and predict voting patterns (e.g., Ninness et al., 2012). The list of growing exemplars from throughout the

academic and scientific community is virtually endless. See the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/) for a multitude of investigations and data sets using neural networking systems.

Within the area of instructional technology, there has recently been a renaissance of applications aimed at enhancing educational interventions. To list but a very few examples, neural networking systems have been applied to classifying and predicting student math skills during interactive training (e.g., Ninness et al., 2005), developing student learning models and predicting knowledge (e.g., Desmarais, Meshkinfam, & Gagnon, 2006), adapting computer student assessment and prediction (e.g., Desmarais & Pu, 2005), classifying and predicting student musical skills (e.g., Ninness, et al., 2013), developing instructional systems providing concurrent assessment and tutoring (e.g., Feng, Heffernan, & Koedinger, 2009), developing online personalized learning systems (e.g., Heller, Steiner, Hockemeyer, & Albert, 2006), and analyzing online cognitive tutors (e.g., Aleven, 2013; Koedinger, Corbett, & Perfetti, 2012).

Nonverbal Network Learning

Neural networks also have been developed to model the behavior of lower organisms. For example, Donahoe and Burgos (2000) employed neural network training to identify levels of response strength in terms of accelerated and extinguished operant behavior. Kemp and Eckerman (2001) continued the lower organism learning analogy by exposing well-trained algorithms to various schedules of reinforcement and obtained cumulative records with response patterns very similar to those of nonverbal organisms exposed to the same schedules of reinforcement. Burgos (2007) demonstrated that neural networks were capable of performing in ways that were analogous to autoshaping and automaintenance (cf. Hamilton & Silberberg, 1978).

Emergent Experimental Possibilities

One type of neural networking system, the connectionist model (CM), has demonstrated exceptional potential for simulating human learning in a variety of stimulus equivalence preparations. Specific to the present discussion, ever-expanding behavioral and computer technologies support the development of CMs for exploring a multitude of derived stimulus relations as well as an assortment of control conditions. In some cases, CMs allow the examination of variables that, because of ethical or time-intensive complications, would be challenging to study systematically with human participants. For example, various components of a CM can be disrupted or incapacitated to explore the resulting learning and performance deficits. By removing particular neural connections, adding disturbance (noise to the training data), and introducing inaccurate weights within the algorithm, we can impair the efficiency and accuracy of the model's performance in ways that are somewhat analogous to the impaired performances seen in actual cases involving human neurological injury. For example, if we disturb the connections within a neural network that are capable of reading textual input, the network's resulting reading deficits are very similar to the impaired verbalizations demonstrated by humans with acquired dyslexia as a function of cerebral insult (Bullinaria, 1997).

Computer Simulations of Derived Relational Responding As discussed previously, stimulus equivalence has been a continuously expanding source of laboratory and field investigation. Notwithstanding this development, extraexperimental influences on humans who spend only a small portion of their existence within a behavioral experiment can hamper our ability to achieve the type of rigorous experimental control obtained in nonhuman laboratory investigations (see Critchfield & Fienup, 2013, regarding this particular issue). Lyddy and Barnes-Holmes (2007) suggest that "considering such restrictions on a full understanding of the development of stimulus equivalence, other sources of evidence regarding the effect might warrant consideration. One such source might be afforded by means of computational modeling" (p. 15). In fact, there has been considerable emphasis placed on exploring how artificial neural networks in the form of CMs might further our understanding of derived stimulus relations (e.g., Cullinan, Barnes, Hampson, & Lyddy, 1994; Lyddy & Barnes-Holmes, 2007; Lyddy, Barnes-Holmes, & Hampson, 2001; Tovar & Chavez, 2012; Vernucio & Debert, 2016) as well as neural network applications focusing on highly specialized problems within behavior analysis and related fields (e.g., Burgos, 2007; Donahoe & Burgos, 2000; Ninness, Henderson, Ninness, & Halle, 2015; Ninness et al., 2012, 2013).

Computer Simulations and Contextual Control of Behavior

A distinctly behavior–analytic approach to CM architectures called RELNET has been used to model complex human response patterns. For example, a now-classic study by Steele and Hayes (1991) demonstrated that human responding came under the contextual control of specific types of stimulus relations (i.e., same, opposite, and different) in accordance with their particular training history. In a RELNET simulation of the same experimental phenomena, Barnes and Hampson (1993) successfully replicated the contextual control of derived stimulus relations that were produced by human participants in the experiment conducted by Steele and Hayes. In a broadly similar investigation, Cullinan et al. (1994) simulated stimulus equivalence in conjunction with a transfer of sequence function. Prior to the RELNET analysis, human participants were trained on a two-response sequence function and assessed on the emergence of a novel three-response sequence. Only the participants who received conditional discrimination training demonstrated transfer to novel stimuli consistent with the three-response sequence function. In the parallel simulated experiment, the RELNET model generated response patterns consistent with the performances exhibited by human participants.

Lyddy and Barnes-Holmes (2007) employed RELNET to assess equivalence class formation by way of linear and one-to-many training protocols. Consistent with performances by human participants (e.g., Arntzen & Holth, 1997), RELNET outcomes demonstrated that the one-to-many training strategy required approximately half as much training time as the linear preparation. The findings from this study might be viewed as one means of vetting training strategies prior to running actual human applications. Here, outcomes clearly indicated that the one-to-many training approach was much more efficient than linear training. The results suggest that conducting a CM analysis prior to running human applications provides a means of assessing the

efficiency of a wide range of equivalence training strategies. This study stands out as a compelling example of the robust findings that can be obtained by computer simulation of behavior–analytic tasks. Such outcomes validate the general premise of CMs: that simulated neural networks can be devised that profit from experience in much the same way as the nervous systems of actual human learners.

RELNET studies have also revealed response patterns that might not have been anticipated by the behavior–analytic community. For example, Lyddy et al. (2001) developed an enhanced version of RELNET that, subsequent to training, was able to "mirror" active versus passive voice in the form of simulated sentences. Notably, the trained version of RELNET demonstrated the ability to form completely novel but correctly arranged, untrained response sequences. An untrained or control network was completely unable to perform any of these tasks. Overall, RELNET investigations have shown considerable promise in modeling how stimulus relations are acquired and expressed in humans; however, certain components of the RELNET architecture have become the subject of some theoretical concern. We will address these theoretical concerns later in this article.

CMs and Human Learning

We will now explain how a CM is changed by experience and in the process introduce some of the technical terminology (for a glossary, see Table 1) that is used to discuss CMs. Consistent with the acquisition of new skills by human participants, before a CM can solve novel problems, it must undergo a series of supervised training trials. As described by Ninness et al. (2013), "neural networks must undergo a series of training sessions with training values in conjunction with expected output patterns before they are capable of providing solutions to problems" (p. 52). During this series of training trials, referred to as epochs, an algorithm known as feedforward backpropagation (Rumelhart, Hinton, & Williams, 1986) compares the CM's calculated values with the target values and identifies the magnitude and direction of difference. During each epoch, the network attempts to bring its calculated values closer and closer to the actual or target values. And, during each epoch, the mean square error (MSE) is obtained by averaging the squared deviations between the network's calculated values and the known target values (cf. Cohen & Sackrowitz, 2002). The algorithm continues to reduce the difference between the calculated and target values until a specified number of training epochs is completed or until a sufficiently small MSE value is reached.

After training a CM until it is able to model a set of input values, the algorithm has acquired the ability to function independently in the analysis of similarly structured but previously unseen input values. As an analogy, many people have trained their computers to develop speech recognition. Usually, such training requires several trials before the system becomes consistently accurate in identifying a new user's unique vocal inflections and oral word formations. In time, the system learns to recognize the voice of a particular user, and the network will become even more proficient with increased exposure to the user's enunciation, articulation idiosyncrasies, and unique voice inflections. The same can be said about training stimulus relations by way of a CM neural network; that is, following a series of supervised training trials, the CM becomes capable of independently deriving novel (untrained) stimulus relations even

Table 1 Neural network terms and definitions

Term	Definition
Bias values	Bias values are constant values (usually dummy values of 1) that are added to the computation of hidden and output neurons prior to activation.
Epoch	An epoch is a complete presentation of the entire set of training values to the CM.
MSE	The MSE is a performance function that computes an average measure of training error. It is obtained by continually averaging the squared deviations between the network's current calculated values and the desired target values. Generally speaking, the smaller the MSE becomes during training, the more effectively the network has been trained.
Learning rate	The learning rate is an adjustable setting that controls the rate of change in the weight and bias values during each training epoch. Setting the learning rate too high (too close to 1.0) increases the chances of exceeding the optimal weight and bias values needed to bring about maximum performance in a trained neural network.
Momentum	Momentum operates in conjunction with the learning rate to increase the speed of network learning. Usually, it is necessary to reduce the size of the learning rate when employing a large momentum value. Combining high learning rates and momentum terms (e.g., both values at or near 1.0) increases the likelihood that CM training may generate an inaccurate model.
Test values	Test values are completely unknown to the CM. These values are presented to the CM upon completion of training. During testing, the CM measures the extent to which the trained network is able to correctly derive new stimulus relations (e.g., stimulus equivalence) using values with which the network has no previous direct experience.
Training values	Training values (training data) are composed of input values that are representative of a particular type of problem to be solved by the neural network.
Simulation number	A simulation number is a machine-generated, pseudorandom number employed in the simulation of a new participant that has not been exposed to training. Advancing the simulation number deletes all previous learning and initializes the network to a new set of random weights. This process is analogous to starting the training protocol for a new, experimentally naive participant.

MSE mean square error, *CM* connectionist model

when being presented with novel stimuli for which it has had no training. This is the phenomenon of CM learning. The network learns a set of relations based on a representative sample of stimuli from a known population, and it becomes capable of generalizing to new stimuli for which it has no previous training. In this way, neural network learning and human learning share a common capability to generalize newly acquired skills to stimuli not previously encountered.

Training Stimulus Relations to Humans and Simulated Humans

Rather than training a neural network to develop speech recognition or solve any number of the aforementioned complex problems, training can be aimed at having a CM neural network identify abstract stimuli as belonging to (or not belonging to) the same class. Because this article explores stimulus relations systems that employ the yes–no training preparation (see Tovar & Chavez, 2012), we will provide a few preliminary details regarding the components of this training protocol as it applies to training human and simulated human participants.

When using a yes–no procedure to train stimulus relations in humans, abstract stimuli (e.g., symbols or abstract academic concepts) are displayed on a computer screen concurrently (e.g., A1B1 are adjacent to one another). During other trials, B1C1 might be displayed side by side on the screen as compound stimuli. The participant identifies the stimuli as belonging to (or not belonging to) the same class by clicking either a *yes* or a *no* button (cf. Nason & Zabrucky, 1988). Prior to training, the participant has no way of knowing which stimuli form class membership, and hence initial responding is unsystematic. During training, selection of the correct buttons produces onscreen reinforcement; yes–no preparations may or may not employ mild punishment contingencies when participants fail to select the correct button.

A CM version of the yes–no preparation can be employed when training stimulus relations. However, in contrast to abstract symbols or abstract academic concepts used when training verbally able human participants, stimuli are encoded as a series of binary activation units that are either on (1) or off (0) when training stimulus relations to CMs. For CM training purposes, this is a way to specify a particular abstract symbol. For example, a white heptagon may be represented in the form of a unique pattern of 1s and 0s where the stimulus is uniquely positioned as a 1 within a row of 0s. The CM might learn group membership with other abstract symbols (e.g., a black scalene triangle) that could be represented in a different pattern of 1s and 0s where this stimulus is uniquely positioned as a 1 within a row of 0s. Clearly, a black scalene triangle has no physical similarity to a distinctively sequenced row of 1s and 0s, but for the purpose of CM training, the actual form of the stimulus does not matter. For neural network training purposes, it is enough that a precisely sequenced row of binary units function in place of a specific abstract stimulus. For every abstract symbol that might be used in training human participants, there is a pattern of binary units that can represent the symbol for a simulated human participant.

Thus, rather than pictures of triangles, circles, or any abstract stimuli, each row of activation units is composed of 1s and 0s that are related or not related to one another. For a CM to acquire discriminations in accordance with correct class membership, the network must learn to recognize the commonly shared patterns that may exist among different rows of binary activation units. Although there may be exceptions and limitations, almost any set of stimuli employed in the training and testing of human participants can be encoded as binary activation units for training and testing by way of a CM (see the Appendix for illustrations of binary stimuli representing abstract symbols).

Derived Stimulus Relations and CM Learning Processes

Neurons within a network that accept input stimuli are referred to as being part of the input layer. Those neurons that transmit values to locations outside the network are referred to as being part of the output layer. Neurons that only interact with other layers within the network are described as residing within the hidden layer (see Fig. 1 for an illustration). In order for a CM to derive stimulus relations, the layers of neurons within the network must be connected to one another. It is through these connections that the layers of neurons are able to interact and learn the relations among stimuli. The connections between layers of a network are "weighted," and these weights have particular values that continually change throughout the learning process. The weights

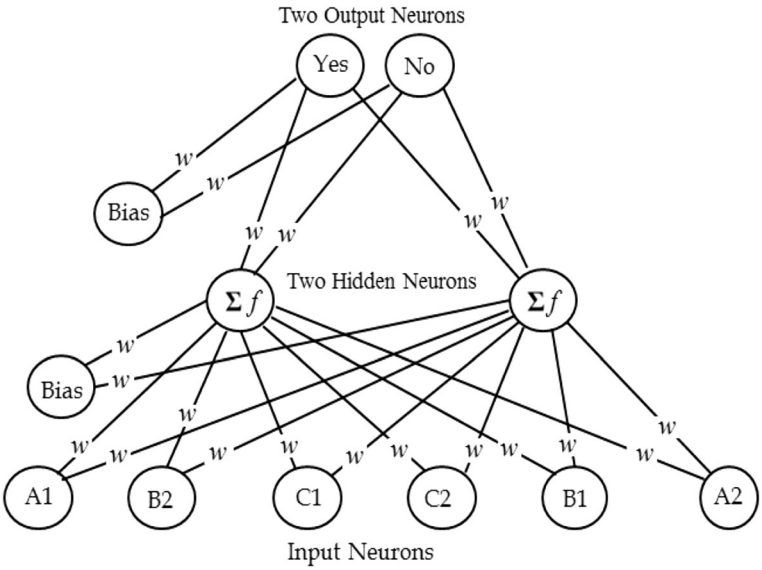


Fig. 1 EVA's architecture includes six neurons in the input layer, two hidden neurons in the second layer, two output neurons in the third layer, and two bias nodes. *EVA* emergent virtual analytics

between connections determine how much the training values change as these values pass along the connections between layers.

Multiple-Exemplar Training From a behavior–analytic perspective, humans learn as a function of direct-acting contingencies in conjunction with verbal rules. In a very general way, the same can be said of CM learning. Virtually all CMs use some type of learning rule in order to adjust the weights in accordance with training input patterns. In a very real sense, CMs learn by coming into contact with multiple exemplars, as do their human counterparts. Just as a child learns to recognize particular spoken and written words by interacting with language from a broader population of verbal exemplars, a CM learns to recognize unique patterns of 1s and 0s (or other numerical configurations) by interacting with representative training patterns from a sample of numerical exemplars that are representative of a broader population.

Just as there are many types of rules that humans may employ within particular contexts during multiple-exemplar training, there are different types of learning rules employed by neural networks for solving specific problems. For the purpose of CMs trained as models that will become capable of deriving stimulus relations, we focus on the application of the delta rule (often referred to as the generalized delta rule). This rule is a primary component of CM models that use feedforward backpropagation as part of their architecture. According to the delta rule, for every input pattern, the results are contrasted with the known or accurate results. To the extent that differences between the calculated and known outputs are identified, the weights and bias values are modified to bring these values closer to one another (bias values are described in the Appendix). Using the delta rule, the CM recalculates the amount of error that exists between its calculated answer and the actual (known) answer, and during each epoch,

the CM makes adjustments in the weights and bias values within each layer of the network.[1] In much the same way as their human counterparts benefit from multiple practice trials, repetitive exposure to training stimuli in conjunction with the delta rule allows the CM's participants to become fluent when responding to entirely new sets of input stimuli—stimuli for which the network has had no previous exposure.

The Delta Rule and the Acquisition of Stimulus Equivalence Another way to conceptualize the delta rule is that this function acts to gradually adjust the connection weights of the CM such that, as a function of consequences, the neural network weights are adjusted and the system becomes increasingly proficient at producing accurate responses to patterns of training stimuli (often referred to as training data). Throughout training, the CM is continually presented with numerical patterns, and the delta rule adjusts the CM connection weights in an attempt to calculate increasingly accurate responses to each input pattern presented to the network.

Because the delta rule engenders the identification and correction of errors, the acquisition of equivalence relations advances exponentially during each training epoch. The delta rule requires that training stimuli go through a series of calculations or processes at each layer and move through a series of forward and backward passes throughout training. During the forward pass (feedforward), training values are delivered to the input layer, where these values are modified and then directed to the hidden layer. Within the hidden layer, the training values are processed again and then transferred to the output layer for further processing. At the output layer, the calculated values are contrasted with the known target values. Because there will not be an exact match between the calculated values and the known target values (particularly at the beginning of training), an error value is computed at the output layer. As an operational definition, CM error is the difference between the calculated values and the target values as determined at the output layer. If the training process advances as intended, the error values should diminish with each epoch. However, as with the training of human participants, during any particular epoch, the network's error level tends to oscillate somewhat.

Backpropagation During what is called the backward pass (or backpropagation of error), the error values (or error signal) return to the neurons within the hidden layer. Here, the neurons update the current error and compute increasingly accurate weights, connecting the neurons in the input layer and the hidden layer. Updating a neuron's current error is comparable to providing consequences for varying degrees of accuracy. With each forward and backward pass of the training values, the weights between layers become increasingly accurate, thus reducing the CM's error and generating better approximations of the target values. With each forward and backward pass, the CM should become better at identifying the relations among stimulus patterns that do (or do not) constitute equivalence classes. If network training is successful, the error value will decrease to near zero as the differences between the calculated and target values

[1] Thus, as will be explained shortly, "feedforward" means that experience with stimuli is passed sequentially through the network's input, hidden, and output layers. "Backpropagation" means that these layers, in turn, are altered by the experience so that they accommodate future experience differently. In broad strokes, the process is similar to feedback loops that are familiar in discussions of human learning.

become smaller and smaller. And, if network training is successful, the CM will have learned to differentiate input training stimuli consistent with stimulus equivalence.

Behavior Has Consequences As mentioned previously, CMs learn as a function of being exposed to multiple exemplars, and as described by McCaffrey (2014), training a CM is fundamentally a process of training simulated participants to acquire the correct weights and bias values that allow the computed output values to match a set of known target values. Epoch after epoch, simulated participants are exposed to correct examples by way of repeated presentations of training stimuli. However, simulated participants are not just "exposed" to the training stimuli; they respond to the training stimuli correctly or incorrectly, and they receive consequences for doing so. Critically, their responses to these training stimuli have consequences in the sense that network neurons are altered by the delta rule during each epoch. The weights are often described as providing varying degrees of "credit" or "blame" to hidden neurons (cf. Hagan, Demuth, & Beale, 2002), depending on their levels of accuracy during each epoch. One might just as easily say that neurons are reinforced or punished in accordance with their performances during each epoch.

As described earlier, during each epoch, the error value fluctuates but decreases as the differences between the calculated and target values diminish in accordance with the delta rule. Analogous to human learning, the network's simulated participants exhibit improving behaviors that are differentially controlled by their consequences, and it is only a mild extrapolation to suggest that the CM's improved behavior has been "shaped as a function of the consequences provided" throughout the duration of training.

Generalization Tests The feedforward and backpropagation processes are repeated until a specified number of epochs are completed. At this point, all training procedures are terminated, and the CM is prepared to receive test stimuli. Consistent with the training and testing of human participants, subsequent to training, the CM is prepared to undergo generalization tests. Importantly, the network will have had no previous exposure to the stimuli used during generalization tests (see the Appendix for a similar but computational description of these same processes).

Human Linguistic Repertoires and Extraexperimental Training Stimuli

When training stimulus relations, it is not unusual for CM software developers or users to enhance their training protocols by employing supplementary (extraexperimental) training stimuli. These extraexperimental training stimuli are added to the primary training stimuli in order to enhance the CM's ability to more efficiently and accurately establish the formation of equivalence classes. The logic for employing extraexperimental stimuli varies among researchers, but in general this is conceived as an attempt to compensate for the presumed pre-experimental repertoires human participants may have "naturally acquired" prior to entering an experimental setting.

For example, in a given experiment addressing derived stimulus relations, human participants might receive training with abstract stimuli in the form of A1B1 and A1C1

in order to derive B1C1. In the same experiment, humans might receive training with stimuli in the form of A2B2 and A2C2 in order to derive B2C2. In a CM experiment with the same ambition, a simulated human is likely to receive supplementary training inputs in the form of XY, YZ, and XZ. In other words, the simulated organism is given a "history" of forming relations involving stimuli other than those in a given experiment, much as all human participants may be expected to bring to bear on any experiment a variety of pre-experimental linguistic repertoires (cf. Vernucio & Debert, 2016). Another way of conceptualizing the use of extraexperimental training is the view that simulated human participants require supplementary training in order to compensate for their lack of familiarity with naturally occurring stimulus relations. But this begs the question: How can adding supplementary or extraexperimental training stimuli to the actual training stimuli reproduce a human's linguistic repertoire? The extent to which extraexperimental training stimuli are actually a critical component to the CM's acquisition of stimulus relations will be explored later in this article.

CMs and Behavior Variability

More than sophisticated statistical algorithms, CMs based on feedforward backpropagation are distinguished by their uncanny abilities to simulate human learning in the sense that they can demonstrate amazing levels of accuracy while, just as human participants, they are subject to fluctuations in learning proficiency. To the extent that a CM has undergone sufficient training, a particular simulation may accurately derive new complex "untrained" stimulus relations. However, just as with human participant training, in which one individual's learning does not fully predict another's learning under identical circumstances, even when a simulated participant produces extremely precise outcomes, subsequent simulations (runs) are likely to generate slightly different accuracy levels. This is because each time the simulation number is advanced, all prior network learning is deleted (in essence, creating a fresh nervous system), and the training of an entirely new simulated participant begins. As in the learning outcomes obtained by different human participants, each new simulated participant should and will show some degree of difference.

This inevitable variability in learning outcomes across simulations has been the source of fascination and some consternation among researchers. From a purely statistical perspective, a neural network often demonstrates exceedingly accurate results even when analyzing chaotic and extremely nonlinear data that would be very difficult to examine by way of conventional statistics; however, neural network models are notorious for obtaining somewhat inconsistent predictions from one simulation to the next (see Haykin, 2008, for a discussion). Such variations often require that a given model be run hundreds (or even thousands) of times before allowing a statistically oriented researcher to have confidence in the accuracy of a prediction. However, from the perspective of a researcher who wants to simulate human learning, neural networks demonstrate human-like variations across simulated participants—that is, CMs exhibit somewhat variable but generally improved task performances with increasing levels of training.

Note that CM learning as described herein makes an implicit assumption regarding one's research agenda. Depending on one's research ambition and orientation, neural network analysis may or may not be a useful methodology. However, the fact that CMs

do not consistently generate identical outcomes during consecutive simulations of human participants makes them ideal "pilot participants" for training and testing stimulus equivalence procedures. In this way, CMs can provide an efficient means of vetting training strategies prior to running actual human applications. As will be discussed, the fact that network simulations of human behavior inevitably produce some inconsistencies across participants is essential to the simulation of human learning.

Replications Without Duplicators

RELNET is unique among neural network architectures in that its first layer includes three hardwired elements, called sample marking duplicators, that in effect preset the network to identify particular types of contextual relations. In effect, these duplicators are hardwired to receive specific types of input values in order to identify contextual relations pertaining to same, different, or opposite. From the perspective of seeking to model learning, this might be considered a network design flaw because sample marking duplicators do not allow the network to acquire contextual relations independently. As RELNET's developers have noted, RELNET preset architectures may provide the network with too much of an advantage regarding what types of stimulus relations it should learn (Barnes & Hampson, 1993).

Recently, CMs have been developed with the ambition of replicating the human acquisition of derived stimulus relations independent of sample marking duplicators. Tovar and Chavez (2012) explained that the goal of their study was "to develop a CM to evaluate stimulus class formation, avoiding the problems generated by the marking of stimulus functions and allowing an appropriate evaluation of CMs capability to respond correctly to emergent relations between stimuli" (p. 749). In their two-part investigation (first human, then computer simulated), these researchers developed an automated yes–no preparation that employed compound stimuli. During each trial, human participants were exposed to two adjacent stimuli and clicked one of two buttons labeled YES and NO. Clicking the correct button produced onscreen points, and clicking the incorrect button resulted in point loss. Another way of conceptualizing this is that participants were trained to click YES in response to within-class stimuli and to click NO in response to between-class stimuli. Employing this protocol, 4 out of 6 human participants exhibited the formation of equivalence classes. As a parallel or analogue experiment within this same study, Tovar and Chavez demonstrated the formation of equivalence classes by way of six computer simulations of the human experiment where 5 out of the 6 simulations exhibited stimulus class formation. Essentially, the CM employed in this study replicated the derived stimulus relations that were produced by human participants. With regard to the findings obtained in their 2012 study, Tovar and Chavez concluded:

> In general, both the human study and the CM showed the relevance of analyzing arbitrary stimulus class formation with different research strategies. Both strategies give each other feedback and generate a reciprocal heuristic value. The use of different experimental and simulation procedures extends our knowledge on the determinant factors involved in arbitrary categorization and symbolic processing. (p. 760)

Vernucio and Debert (2016) conducted a replication of the Tovar and Chavez (2012) study, employing the same training and test data; however, these authors conducted their CM analysis using a go/no-go preparation such that the CM protocol only made yes responses when presented with related stimuli. Outcomes demonstrated that 4 out of 6 simulated participants formed stimulus relations consistent with equivalence. Thus, the Tovar and Chavez (2012) and Vernucio and Debert (2016) studies provide elegant examples of CMs that are capable of simulating human learning of derived stimulus relations without the assistance of sample marking duplicators. Yet neither of these studies attempted to demonstrate *contextual control* of conditional relations (e.g., opposition, distinction, greater than, or less than). Therefore, it might be premature to dismiss RELNET as a foundational architecture for simulating the acquisition of contextual control of derived stimulus relations.

Remodeling CMs

The discussion thus far may be summarized as follows: All of the CMs described in this article are powerful in the sense that they are able to simulate derived stimulus relations that are simply not possible to achieve by employing traditional statistical algorithms or by training nonverbal organisms. At the same time, virtually all current computer learning models are in a state of continual revision and remodeling, and the feedforward backpropagation algorithm is no exception (see McCaffrey, 2014, for a discussion). In the CMs employed by Cullinan et al. (1994), Lyddy et al. (2001), Lyddy and Barnes-Holmes (2007), Tovar and Chavez (2012), and Vernucio and Debert (2016), supplementary input values XY, YZ, and XZ were added to the primary training values to mimic the participants' pretraining (extraexperimental) history. As noted previously, extraexperimental training values are employed to compensate for the CM's unfamiliarity with stimulus relations generally and thereby to augment the CM's ability to form class membership. However, a more rigorous approach to CM simulation of human behavior might preclude training with such extraexperimental training values—that is, if a CM is capable of running simulated participants without the advantage of supplementing training data with extraexperimental values, this would appear to be a more parsimonious strategy for modeling the human acquisition of stimulus relations. In the remainder of this article, we describe emergent virtual analytics (EVA) as a CM intended to simulate derived stimulus relations without the incorporation of sample marking duplicators and independent of any types of extraexperimental training values.

The Evolution of EVA

As described previously and shown in Fig. 1, in CMs, input and bias values are first transformed by being multiplied by a set of small randomized weights (w). Then, these weights and bias values are summed ($\sum$) and activated (f) within the hidden layer and passed to two neurons located within the output layer. Upon receiving the values forwarded by the neurons within the hidden layer, the output neurons compare the current calculated values with the known target values. Subsequently, the backward pass (i.e., backpropagation) from the output neurons apportions error back to

each neuron in the hidden layer (and then the input layer) in accordance with the current level of error or accuracy that the neuron has produced.

Although EVA is still in the early stages of development, this CM is able to operate with the same training values employed within the Tovar and Chavez (2012) and Vernucio and Debert (2016) studies. However, unlike any of the CMs described thus far in this article, EVA functions without employing extraexperimental training values (i.e., XY, YZ, and XZ) that are meant to compensate for a CM's lack of a pre-experimental linguistic repertoire that exists within human participants. EVA's architecture has six neurons in the input layer, two neurons in the second (hidden) layer, and two neurons in the output layer. As shown in Fig. 1, EVA's architecture includes two bias nodes. For the purpose of illustrating a systematic replication of the Tovar and Chavez (2012) study, we have configured EVA as a variation on a yes–no training protocol based on the backpropagation coding structure as originally developed and described by McCaffrey (2015; refer to Haykin, 2008, for a discussion).

Educating EVA

Conducting a CM analysis with EVA requires only a few input settings and button presses. As shown in Fig. 2, we have set the number of columns at 8 because there is a total of eight columns of input variables, including two target variables, 0 and 1. The number of rows for training data is set to 8 because there are eight rows representing a unique input pattern for each presentation to the network. We have set the number of input neurons at 6, indicating the number of independent variables. Moving to the far right of Fig. 2, momentum is set at 0.10. On the left side of this form, we have set the number of hidden neurons at 2 and the number of rows for test values at 4. The number of output neurons is set at 2, and the learning rate is set at 0.50. Several of these input

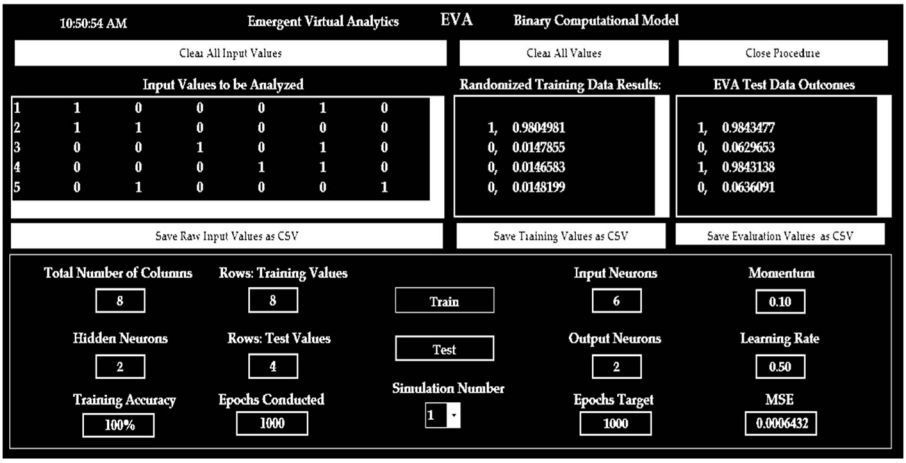


Fig. 2 Windows form allowing interactive training and testing of simulated participants. Raw binary input values are displayed under the "Input Values to be Analyzed" heading. Training results are shown in the center under "Randomized Training Data Results," and test outcomes are shown under "EVA Test Data Outcomes." *EVA* emergent virtual analytics

settings are required because they match the parameters associated with the input file. Specifically, the number of columns, rows, input neurons (independent variables), and output neurons (target variables) must match the input file parameters of the training values data set. A beta version of EVA with sample data sets is currently available as a download for interested researchers (see the Appendix for downloading and configuration details).

Other parameters on the EVA Windows form can be set in accordance with the researcher's preference. The simulation number is a randomization seed value employed in the process of introducing a new participant to training; that is, advancing the simulation number deletes all previous learning and initializes the network to a new set of random weights. This process is analogous to restarting the entire training protocol for a new, experimentally naive participant. The other input values displayed in Fig. 2 are the result of conducting an analysis of the training data when the number of epochs is set to 1000 and the simulation number is set to 1. As mentioned earlier, the MSE is the average value of the totaled squared deviations between the computed outputs and the actual or target values. During the series of training epochs, squaring these differences provides an index of the network's error value. The MSE is one of the most commonly employed measures used to identify overall network training accuracy; however, by itself, it is not a sufficient index of the network's forthcoming performance on previously unseen or new test values.

Binary Encoding of Training and Test Stimuli

As described earlier, unlike abstract symbols or complex academic material used in the training of human participants, network training stimuli are encoded as binary units (digits) such that each unit within a given row of data (vector) is on (1) or off (0). For all practical purposes, the computer has only two digits with which it must perform all operations. Nevertheless, a row of binary digits in various configurations can represent almost any type of symbol. Note that this is not the same as using the binary numeral system for mathematical operations. As mentioned previously, it is a way of replacing a particular symbol (e.g., a black obtuse triangle) with a unique row of 1s and 0s where the stimulus is uniquely positioned as a 1 within a row of 0s. Obviously, these two stimuli bear no physical similarity to one another; however, for the sake of network training, the appearance of the stimulus is irrelevant. It is sufficient that a uniquely arranged row of binary units functions in place of an abstract stimulus. As mentioned earlier, throughout training, the network detects the common patterns that exist among different rows of binary activation units. Although there are always exceptions and limitations (see Fodor & Pylyshyn, 1988, for an early discussion), virtually any type of visual stimulus or mathematical expression that is used in equivalence-based human training could be parsed as a set of binary digits for neural network training. We provide additional details regarding the encoding of training and test stimuli, the momentum, the learning rate, and related network parameters within the Appendix.

EVA Outcomes

Analogous to skill acquisition by human participants, how well a CM derives stimulus relations depends on how much training is provided. When EVA's training is limited to

only 200 epochs, the model performs in much the same way as an inadequately trained human participant. With this limited number of epochs, EVA completely fails to discriminate between between-class and within-class group membership. As shown in Fig. 3, with only 200 training epochs conducted, EVA is unable to demonstrate any approximation of derived stimulus relations, and task performances with test values reveal that within-class stimuli (A1C1 and A2C2) and between-class stimuli (A1C2 and A2C1) all fall between 0.5717006 and 0.5720407.[2] In this run, the MSE value is 0.4936787, indicating an error rate of nearly 50%. Thus, when the number of training epochs is set at 200, the network's ability to derive stimulus relations is severely curtailed, and the likelihood of EVA identifying class membership is at chance levels. One might say that when the number of epochs is set at or below 200, EVA performs as a tabula rasa.

As shown in the bottom panel of Fig. 3, when the number of training epochs is increased to 500, EVA begins identifying stimuli consistent with equivalence class formation. Looking at within-class stimuli, the task performance levels are now 0.9674799 for A1C1 and 0.9673346 for A2C2. With regard to stimuli that do not belong to either class, the between-class stimuli fall to 0.1143698 as shown under A1C2, and A2C1 falls to 0.1163644. Here, the MSE drops to 0.0034873. Although EVA is not completely trained, setting the number of epochs at 500 rather than 200 allows EVA to begin identifying within-class and between-class stimuli.

As shown in the top panel of Fig. 4, increasing the number of training epochs to 1000 allows EVA to identify the formation of within-class and between-class stimuli with an increased level of precision. Specifically, task performance levels are now 0.9843477 for A1C1 and 0.9843138 for A2C2. With respect to stimuli that do not show class membership, task performance is now 0.0629653 for A1C2 and 0.0636091 for A2C1, and the MSE falls to 0.0006432. The bottom panel of Fig. 4 shows that increasing the number of training epochs to 10,000 continues to have a conspicuous impact on EVA's ability to identify stimuli consistent with between-class and within-class group membership. With this large number of training epochs, we obtain within-class task performance levels of 0.9960288 and 0.9960147 for A1C1 and A2C2, respectively. In turn, we obtain between-class task performance levels of 0.0195751 and 0.0196606 for A1C2 and A2C1, respectively, and the MSE shrinks to 0.0000379.

CMs and Behavior Variability Revisited In the bottom panel of Fig. 3, where 500 epochs are employed during training, the within-class and between-class differences become apparent over a series of 10 separate and consecutive simulations. In the first simulation, the within-class task performances for A1C1 and A2C2 are 0.9674799 and 0.9673346, respectively. However, when employing only 500 training epochs, subsequent task performance levels among simulated participants become conspicuously dissimilar from one another, fluctuating between 0.9300 and 0.9800 (rounded) for A1C1 and A2C2 in turn. With this limited number of epochs, the same level of inconsistency becomes apparent for between-class outcomes.

In the bottom panel of Fig. 4 that shows 10,000 training epochs, the within-class performance level for A1C1 is identified as 0.9960288, and the simulated task performance

[2] Note that we also conducted tests for symmetry; however, these outcomes are not described herein, as the outcomes were entirely consistent with those of our tests for equivalence.

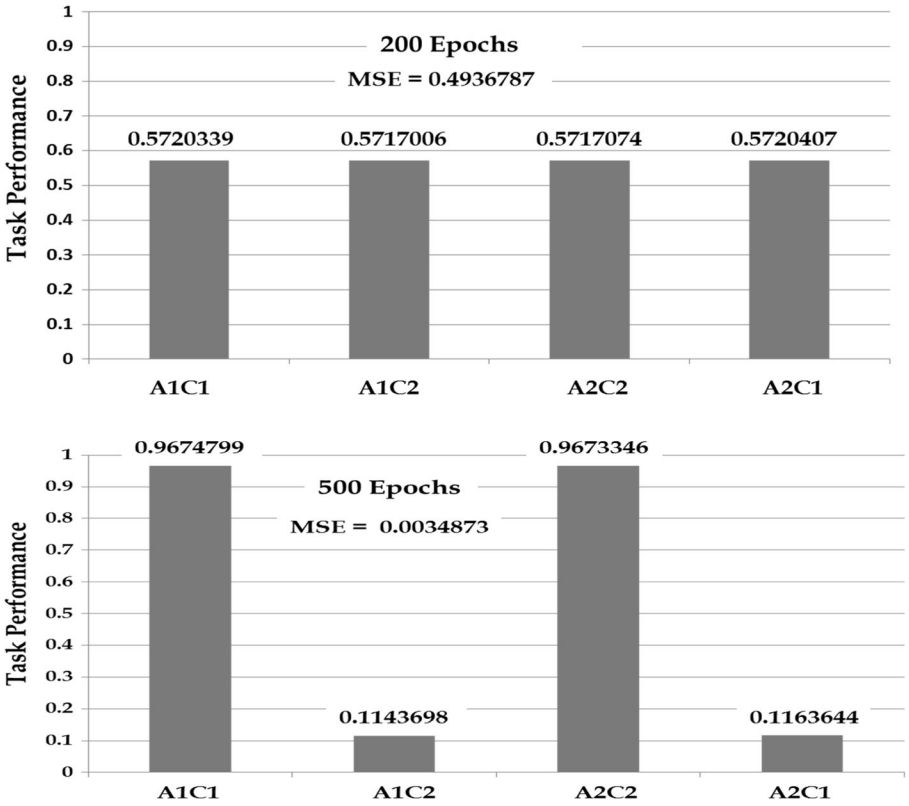


Fig. 3 A1C1 and A2C2 represent task performance levels for within-class stimulus relations, and A1C2 and A2C1 represent task performance levels for between-class stimuli. In the top panel, the connectionist model (CM) analysis is conducted using 200 epochs. In the bottom panel, the analysis is conducted with 500 epochs. *MSE* mean square error

level of A2C2 is 0.9960147. At this level of extensive training, only minor inconsistencies can be seen with respect to the between-class task performance levels shown for A1C2 and A2C1. When 10 separate simulations are conducted at 10,000 epochs each, task performances range between 0.9959 and 0.9962 (rounded) in the formation of within-class stimulus relations. In the same set of 10 different participant simulations, between-class task performances fluctuate between 0.0193 and 0.0197 (rounded). Clearly, conducting simulations that require only 500 epochs will yield outcomes that are much less reliable than those for simulations conducted at 10,000 epochs.

Consistent with the acquisition of new skills by human participants, CMs exhibit slightly variable but progressively improved task performances with intensified levels of training. In this example, when we increase the maximum number of epochs beyond 10,000, it becomes difficult to identify visually conspicuous changes in performance levels. However, at extremely high training levels not shown or recommended herein (e.g., 100,000,000 epochs), there is a risk associated with overtraining the network. As it turns out, overtraining often results in a problem commonly referred to as "overfitting," a circumstance in which the CM performs extraordinarily well on the training data but may lose its ability to generalize beyond the training data.

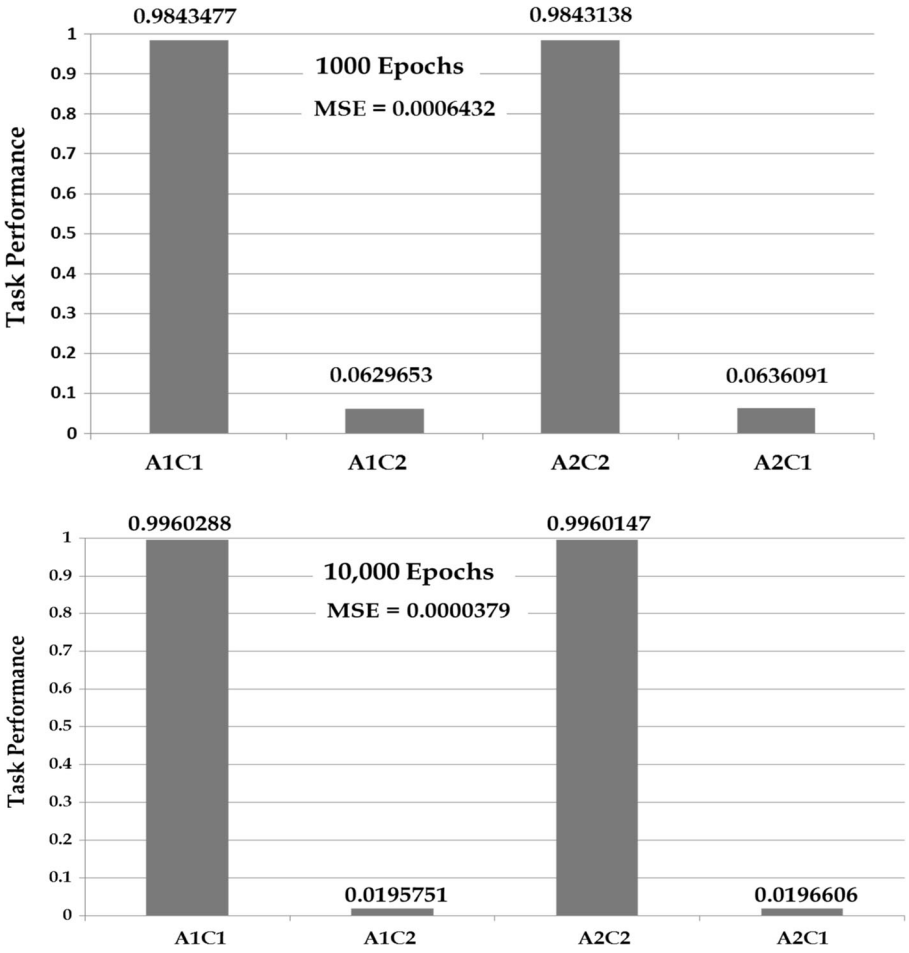


Fig. 4 A1C1 and A2C2 represent task performance levels for within-class stimulus relations, and A1C2 and A2C1 represent task performance levels for between-class stimuli. The top panel shows outcomes obtained after 1000 training epochs. In the bottom panel, the number of training epochs is increased to 10,000. *MSE* mean square error

Training and Overtraining Although the current beta implementation of EVA is far from complete, increasingly accurate levels of equivalence class formation become apparent as we increase the number of required training epochs (e.g., from 200 to 10,000). This gradual transition from poorly differentiated to well-differentiated task performances by the CM illustrates one of the fundamental ways in which neural networks can contribute to our understanding of the human acquisition of stimulus relations; that is, when humans have limited exposure to training trials, they exhibit a reduced likelihood of deriving complex stimulus relations. With increased training, human participants and simulated human participants demonstrate improved levels of task performances; however, there is a training threshold beyond which we often see diminishing proficiency levels when simulating a participant's ability to generalize from training stimuli to novel test stimuli (see Haykin, 2008, for a detailed discussion).

The extent to which we might encounter a similar phenomenon (i.e., diminishing proficiency) when teaching stimulus relations to humans remains an empirical question.

Piloting the Possibilities

Vernucio and Debert (2016) argued for continued CM investigations that analyze class formation by way of the go/no-go procedure with an eye toward demonstrating contextual control of derived stimulus relations. These authors suggested that "future studies could evaluate the possibility of adapting the present model for the simulation of behaviors involving contextual control. So far, no alternatives to RELNET (Barnes & Hampson, 1993) exist for the simulation of such behavior" (p. 448). As we discuss in the following, we believe that there are several forthcoming investigations directed at simulating contextual control of human behavior.

EVA and Contextual Control

To stay within the scope of this article, we have confined our exemplars and our primary discussion to training and testing CMs within the stimulus equivalence paradigm. Among other areas of derived relational responding, we now are examining frames of opposition, distinction, comparison, hierarchy, and other forms of contextual control (Hayes et al., 2001). For instance, the current beta version of EVA is able to form stimulus relations consistent with frames of opposition. In this type of experimental preparation, if EVA is trained such that A1 is the opposite of B1 and A1 is the opposite of C1, then, during testing, B1 and C1 will emerge as being the same. Another way of conceptualizing this is if A1is encoded as 1and B1 is encoded as −1, these training values represent opposites of one another when viewed as locations along the coordinate axes. Similarly, if C1 is encoded as −1, then C1 represents the opposite of A1 when the values are displayed on the coordinate axes. In testing a preliminary example of contextual control of derived stimulus relations, we should find that EVA identifies the opposite of an opposite as being the same. This does not appear to be a particularly challenging task when running simulated participants in EVA, and arranging input stimuli in accordance with the aforementioned description will demonstrate that EVA will identify the opposite of an opposite as being the same. In conjunction with the aforementioned primary training stimuli, we provide sample data sets that allow interested researchers to verify these contextually controlled procedures and outcomes (see the Appendix for details). More complex versions of contextual control will require the evaluation of several types of derived relational responding.

Reciprocal Heuristic Value

As mentioned earlier, Tovar and Chavez (2012) called for an expansion of CM research, indicating that, together, human and neural network investigations show "the relevance of analyzing arbitrary stimulus class formation with different research strategies" and that "both strategies give each other feedback and generate a reciprocal

heuristic value" (p. 760). As a prospect for continuing research and related software development, we continue to expand our investigations of neural network functionality in coordination with human investigations of derived stimulus relations. The current version of EVA represents a preliminary sample of our experimental possibilities. Our ambition is to develop a series of studies comparing and contrasting the ways in which computer models are capable of predicting and reproducing derived stimulus relations as observed in human participants. The extent to which we are able to reproduce highly diversified task performances as seen among humans remains an empirical question.

Caveats

A central feature of the feedforward backpropagation algorithm is its ability to derive solutions in a manner that simulates human problem solving, particularly when being exposed to novel forms of complex input values. As mentioned earlier, all of the CMs described in this article are powerful in the sense that these algorithms are capable of solving problems or making precise predictions that are simply not possible by way of traditional statistical algorithms; however, this power comes with a caveat. As described by Ninness et al. (2015):

> Predictions generated by neural networks are not accompanied by known margins of error. That is, when a network algorithm forecasts a score or a series of scores, behaviors, or classifications, these predictions are not supplemented by values indicating a bandwidth of accuracy within which subsequent results will fall within a known margin of error. (p. 176)

Moreover, for most behavior analysts, encoding training stimuli as binary inputs rather than abstract symbols or academic terms or concepts may not be an intuitively obvious data transformation. Nevertheless, the reviewed studies and tasks simulated herein provide evidence that CMs are capable of performing in ways that are very similar to those seen among human participants (see the Appendix for details regarding encoding training and test stimuli as binary inputs).

CMs and Highly Diversified Behavior–Analytic Tasks

CMs are one of the most quickly evolving technological possibilities throughout the scientific community. Lyddy et al. (2001) argued that CMs offer

> a useful and psychologically plausible simulation of stimulus equivalence and transfer of function phenomena. The results suggest that using connectionist networks and behavior analytic tasks together has potential for advancing the contribution of both approaches to the study of human language and cognition. (p. 426)

With the exponential advances in hardware development, particularly with respect to the current memory capacity and processing speed of modern laptop computers, we have found that students in behavior analysis and related disciplines have become increasingly proficient at developing and using neural network systems.

Going forward, it will be important to study the ways in which CMs can simulate human acquisition of diversified stimulus relations by comparing several types of outcomes: emergent relations that are derived by stimulus equivalence and emergent relations that are derived by contextual control; emergent relations derived by way of MTS procedures versus emergent relations derived by way of go/no-go and yes–no procedures; and emergent relations that can be derived by CMs but that cannot be derived by human participants and vice versa. With the continuing advances in computer learning across the sciences, the extent to which newly developed CMs are able to facilitate our understanding of human learning remains an inevitable question. Answering this question is likely to play a central role in the advancement of derived stimulus relations technology.

Compliance with Ethical Standards No animals or humans were involved in the development of this study. All data were acquired by way of artificial intelligence systems.

Conflict of Interest All authors declare "No conflicts of interest."

Appendix

Conducting a CM Analysis with Training and Test Stimuli

As mentioned earlier in this article, the idea of converting abstract stimuli (e.g., pictorial symbols, mathematical expressions, or verbal concepts) into binary values may seem like an unusual way to simulate stimuli for use in training stimulus equivalence procedures. The binary encoding issue involves the following feature: Rather than using pictorial symbols or conceptual stimuli, which are often employed when training stimulus relations to human participants, stimuli are presented to neural networks as a series of binary activation units that can be either on (1) or off (0). So, rather than pictorial or conceptual stimuli that are related or not related to one another in the training of human participants, each row or vector of activation units is composed of 1s and 0s that are related or not related to one another. During most traditional equivalence-based studies, human participants derive stimulus relations by learning the mutually shared patterns among stimuli. Analogously, in order for a neural network to derive stimulus relations, it must learn the mutually shared patterns of binary activation units. As a practical matter, virtually any set of input patterns of stimuli that can be employed in the training and testing of human participants can be parsed into binary activation units for training and testing by way of a computer model.

Trained Stimuli

Figure 5 displays rows within the exemplar trained stimuli wherein each row contains six unique stimulus arrangements followed by two target variables. The first row—A1B1—displays the digit 1 in columns A1 and B1; the other columns in this row are

		Trained						Tested	
		A1	B2	C1	C2	B1	A2	YES	NO
Trained	A1B1	1	0	0	0	1	0	1	0
Trained	A1B2	1	1	0	0	0	0	0	1
Trained	B1C1	0	0	1	0	1	0	1	0
Trained	B1C2	0	0	0	1	1	0	0	1
Trained	A2B2	0	1	0	0	0	1	1	0
Trained	A2B1	0	0	0	0	1	1	0	1
Trained	B2C2	0	1	0	1	0	0	1	0
Trained	B2C1	0	1	1	0	0	0	0	1
Tested	A1C1	1	0	1	0	0	0	1	0
Tested	A1C2	1	0	0	1	0	0	0	0
Tested	A2C2	0	0	0	1	0	1	1	0
Tested	A2C1	0	0	1	0	0	1	0	0

		Trained						Tested	
		A1	B2	C1	C2	B1	A2	YES	NO
Trained	A1B1	●	0	0	0	▛	0	1	0
Trained	A1B2	1	1	0	0	0	0	0	1
Trained	B1C1	0	0	▼	0	▛	0	1	0
Trained	B1C2	0	0	0	1	1	0	0	1
Trained	A2B2	0	1	0	0	0	1	1	0
Trained	A2B1	0	0	0	0	1	1	0	1
Trained	B2C2	0	1	0	1	0	0	1	0
Trained	B2C1	0	1	1	0	0	0	0	1
Tested	A1C1	●	0	▼	0	0	0	1	0
Tested	A1C2	1	0	0	1	0	0	0	0
Tested	A2C2	0	0	0	1	0	1	1	0
Tested	A2C1	0	0	1	0	0	1	0	0

Fig. 5 The top panel displays the trained and tested stimulus relations presented to emergent virtual analytics (EVA) as an analogue to human–computer interactive learning. In the bottom panel, selected abstract stimuli, as might be seen by human participants during interactive training, are superimposed on the binary training stimuli

set to 0. In binary activation format, this row indicates that the A1 and B1 stimulus units are on (1), whereas B2, C1, C2, and A2 are off (0).

We identify the target variables in the last two columns (within-class stimuli or between-class stimuli) as YES (1) and NO (0), respectively (cf. Tovar & Chavez, 2012). To make the group membership patterns somewhat more salient (as might be seen by humans), we have shaded all stimulus units that share within-class membership. However, unlike the Tovar and Chavez data set, no extraexperimental pretraining input values (i.e., XY, YZ, and XZ) are included within the trained stimuli.

In the bottom panel of Fig. 5, selected abstract stimuli, as might be seen by human participants during interactive training, are superimposed on the binary training stimuli. On particular trials, A1B1 are displayed adjacent to one another on the computer screen. Likewise, on various trials, B1C1 are displayed side by side on the screen. Throughout training, contingent reinforcement is provided for correctly identifying A1B1 and B1C1 stimuli as being members of the same class (i.e., within-class stimuli). During testing, the emergence of A1C1 as being members of the same class indicates the formation of the equivalence class A1B1C1.

Tested Stimuli

With regard to the tested values displayed in the top panel of Fig. 5, there are four rows, and each row contains six unique stimulus units followed by the target variables. For

example, the first row—A1C1—displays the digit 1 in columns A1 and C1; the other columns in this row are set to 0. In binary activation format, this row shows that the A1 and C1 stimulus units are on (1), whereas B2, C2, B1, and A2 are off (0). The first row stands in contrast to all other rows within the tested stimuli. All of the YES column variables display 1s or 0s in order to allow the researcher to visually identify the desired or target values. Note, however, that the last column labeled NO contains only 0 values. Subsequent to training, this last column will display the simulated participants' percentage of correct task performances.

Because comma-separated values (CSV) files can be used with any spreadsheet program, we use this format as an exemplar for the current training data (see the CSV files identified as "Train_8_rows_8_cols" and "Test_4_rows_8_cols" in the supplemental materials). As described within the main text, we also provide two additional data sets (identified as "Train_Opposites_8_rows_8_cols" and "Test_Opposites_4_rows_8_cols" in the supplemental materials) that the user can access and run by way of the EVA network. Figure 6 shows the same training input values displayed in Fig. 5 after being converted to CSV format. Note that all row and column descriptors have been deleted when saving the data to this format. Eliminating descriptors is essential for inputting training and test values.

Running the EVA Application

In Fig. 7, we provide an illustration of the EVA Windows form where all input parameters are prepared to receive the training and test files. As described within the text, the total number of columns and rows for the training values is set to 8 because these values correspond to the number of units within the training file. The number of input neurons is 6 because the first 6 out of 8 column variables in each row represent stimulus units. We set the momentum option to 0.10 in this example. Moving to the left side of this Windows form, we have set the number of hidden neurons at 2. The number of rows for the test data is set at 4 because this represents the total number of rows employed in the test data, and the learning rate is set at 0.50. The simulation number, displayed as 1, is the program's current randomization seed. Advancing the simulation number randomizes all previous training weight and bias values. This is analogous to initiating training and testing for a new human participant; that is, each time the simulation number is advanced, all previous learning is deleted and the input values are randomized (resequenced) by the Fisher–Yates shuffle algorithm. Thus, with each of the 10 possible simulation numbers, EVA initiates the training of a completely new simulated participant.

1	0	0	0	1	0	1	0
1	1	0	0	0	0	0	1
0	0	1	0	1	0	1	0
0	0	0	1	1	0	0	1
0	1	0	0	0	1	1	0
0	0	0	0	1	1	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	0	1

Fig. 6 Training input values (as displayed in the top panel of Fig. 5) converted to comma-separated values (CSV) format

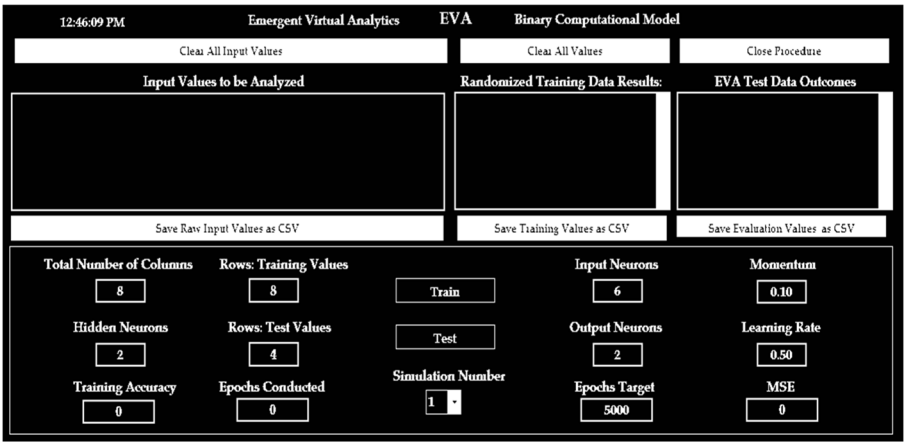


Fig. 7 Emergent virtual analytics (EVA) Windows form employed in the training and testing of simulated participants

Sensitive Settings Virtually all CMs that employ feedforward backpropagation algorithms have several sensitive settings to which the researcher must be attentive. Two of the most sensitive are the learning rate and the momentum. When conducting a series of simulations, the learning rate and momentum settings must remain constant when comparing the task performances across simulated participants. Likewise, the number of hidden neurons employed during training is a parameter that should not be changed when comparing performances across simulated participants.

CM Training Clicking the *Train* button in the center of Fig. 8 opens a new window that allows the user to select the training data. Upon selecting the training file, the CM immediately initiates 5000 training epochs. When training is finished, the number of epochs completed in the training of a simulated participant appears on the left side of

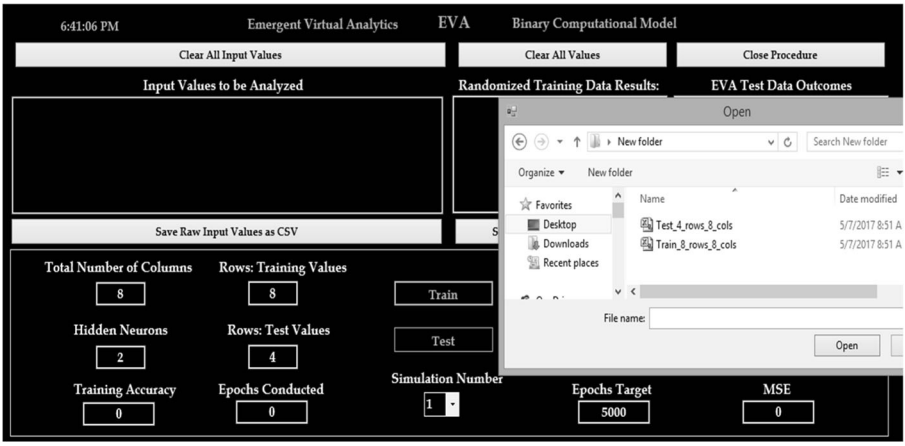


Fig. 8 Emergent virtual analytics (EVA) Windows form showing the settings employed during the training and testing of a simulated participant

the simulation number under the "Epochs Conducted" heading. At this point, this number should match the number displayed under the "Epochs Target" heading.

As shown in Fig. 9, training accuracy is identified at 100%, and the MSE is 0.0000804 in this particular example. Training accuracy and the MSE are important, but these are preliminary metrics of how well the network will perform on test data. These preliminary outcomes suggest that the researcher is well positioned to conduct a CM analysis of the test data.

Training Outcomes Under the "Randomized Training Data Results" heading, the leftmost number indicates the correct (target) value, and the adjacent number indicates EVA's percentage of correct task performances for matching this target (i.e., EVA's predicted value appears immediately to the right of each target value). For example, the first target value in this listing is 1, and the adjacent correct task performance is 0.992878. The second target value is 0, and the adjacent task performance value is 0.0053285. Inspecting all of the outcomes under this heading, it becomes apparent that all of the predicted values are very good approximations of their respective targets. Thus, the researcher is well positioned to conduct an analysis of the test values. Because EVA saves a copy of all weight and bias values calculated during training, the researcher can move directly toward conducting an analysis of the test values.

Conducting an Analysis of the Test Values

Figure 10 displays the test data set after being converted to CSV format. Again, the last column includes only 0s because these task performance values have yet to be calculated by the neural network (see the "Test_4_rows_8_cols.csv" file in the supplemental materials).

As shown in Fig. 11, clicking the *Test* button located directly below the *Train* button opens a new window, allowing the user to select the test values file. When the user clicks the test file name, EVA immediately runs an analysis of the selected test values.

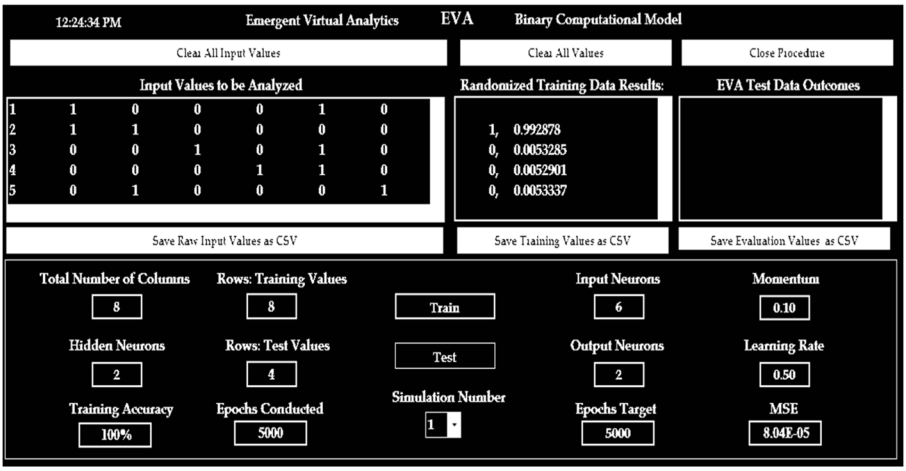


Fig. 9 Emergent virtual analytics (EVA) Windows form displaying training outcomes

1	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0
0	0	0	1	0	1	1	0
0	0	1	0	0	1	0	0

Fig. 10 Illustration of the test data set displayed in Fig. 5 after being converted to comma-separated values (CSV) format

Test Findings

On the far right side of Fig. 12, the test data findings appear under the "EVA Test Data Outcomes" heading. Within each row, there are two values. The target values of 1 or 0 appear to the left of their corresponding percentages of correct task performances. For example, the first target value in this listing is 1, and the adjacent task performance value is 0.9942561.The second target value is 0, and the adjacent task performance value is 0.0269981. Clicking the *Save Evaluation Values as CSV* button, the user is able to save a copy of the test outcomes to any computer location.

Opening the saved CSV file, the network analysis of the test outcomes appears as shown in Fig. 13. The target values, YES (1) and NO (0), fall under column A, and the network's calculated percentages of correct task performances appear immediately adjacent to each target value under column B. The results can be converted into a bar graph by highlighting the column B data and inserting a column chart.

Based on the results displayed in Fig. 13, Fig. 14 shows a bar graph of the network's test findings. Each bar depicts how well the network performed with respect to deriving stimulus relations for each of the four possible classifications. A1C1 and A2C2 refer to the emergent stimulus relations for within-class membership, whereas A1C2 and A2C1 refer to stimuli that do not form class membership (between-class stimuli).

As shown in Fig. 14, setting the maximum number of epochs at 5000 allows the model to identify stimuli that are members of the same class. Training the network with this fairly large number of epochs, the MSE falls to 0.0000804, and the within-class and between-class stimuli are well differentiated where the task performance levels are 0.9942561 for A1C1 and 0.9942381 for A2C2. As might be expected, Fig. 14 shows

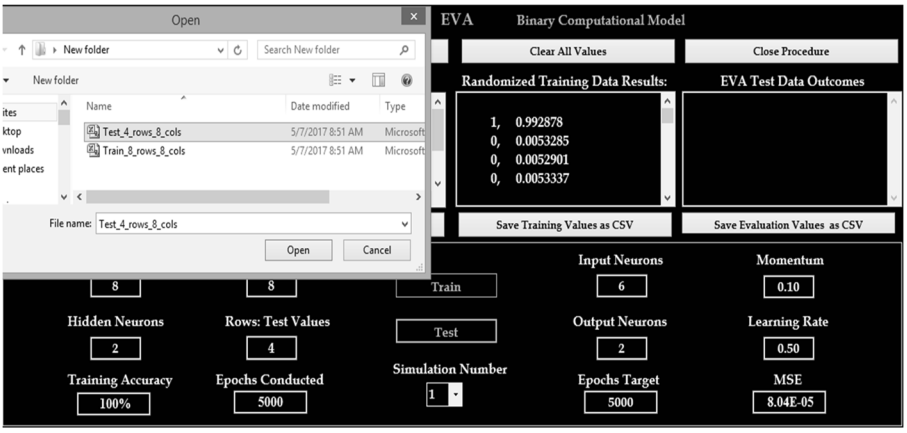


Fig. 11 Illustration of the emergent virtual analytics (EVA) Windows form allowing the selection of the test file

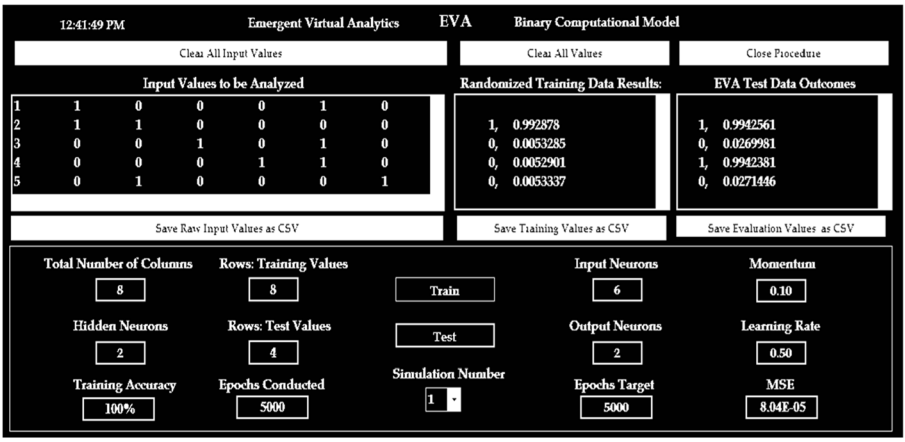


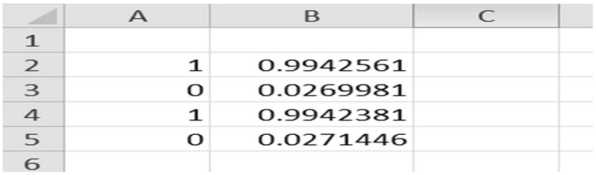
Fig. 12 Test data findings appear under the "EVA Test Data Outcomes" heading. *EVA* emergent virtual analytics

the between-class stimuli outcomes at extremely low task performance levels; that is, the stimuli in these classes did not form group membership.

Computational Processes As described earlier, for all input values presented to the CM, the algorithm generates a series of randomized weights (between −1 and 1 or much smaller) that function as multipliers between neurons. Subsequent to multiplying these input values by these randomized weights, the values are summed and passed to neurons located within the hidden layer. Within the hidden layer, the algorithm transforms all weighted inputs and bias values using an activation function (e.g., the hyperbolic tangent function) and then passes these values along to the output layer for comparison with the known target values.

As a practical matter, bias values are constants (usually values of 1) that act to shift all inputs away from the center (origin) of the coordinate axis. Oftentimes, bias nodes are not discussed or illustrated in streamlined descriptions of neural network algorithms; however, they are essential components of the feedforward backpropagation algorithm. As stated by McCaffrey (2014):

> Training a neural network is the process of finding a set of good weights and bias values so that the known outputs of some training data match the outputs computed using the weights and bias values. The resulting weights and bias values for a particular problem are often collectively called a model. The model



	A	B	C
1			
2	1	0.9942561	
3	0	0.0269981	
4	1	0.9942381	
5	0	0.0271446	
6			

Fig. 13 Spreadsheet illustration of the network's test findings

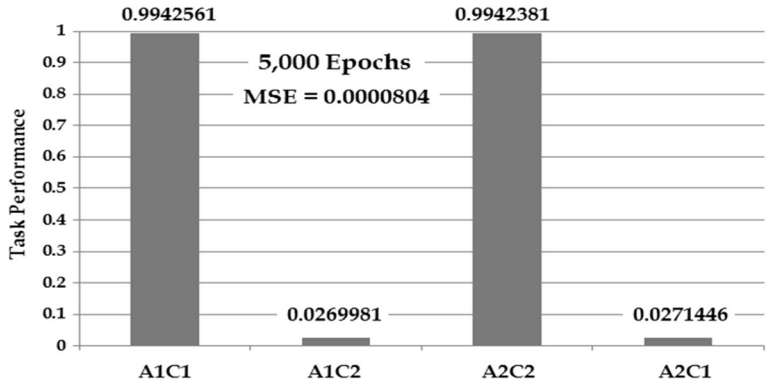


Fig. 14 A1C1 and A2C2 represent connectionist model (CM) task performance levels for within-class relations, and A1C2 and A2C1 represent the network's identification of stimuli that fall between classes. *MSE* mean square error

> can then be used to predict the output for previously unseen inputs that do not have known output values. (p. 95)

Figure 15 provides a didactic illustration of the multifaceted feedforward backpropagation algorithm in the form of a flowchart. In this illustration, the input and bias values enter at the top of the chart and are passed downward and then recycled during backpropagation; that is, the weight and bias values are continually reprocessed to gradually improving levels of accuracy over a series of forward and backward passes.

To revisit the flow of learning originally shown in Fig. 1 (but with inputs positioned at the top of this illustration), the weight and bias values are summed and fed forward (solid arrow lines) to the activation function in the hidden layer (the hyperbolic tangent function). Then, new weight and bias values are calculated and forwarded to the output layer, where the values are compared with the target values (the Softmax function).

Upon completing the forward pass to the output layer, the backpropagation process begins. Each backward pass (dashed arrow lines) from the output layer recalculates the weight and bias values in accordance with the current level of error that a neuron has produced and returns these values to the hidden layer for processing. The hidden layer applies the activation function and returns the updated values to the input layer. The feedforward and backpropagation processes continue until a specified number of epochs are completed. When the required number of epochs has been accomplished, the training process is stopped, and the researcher can run his or her test data based on a trained CM (see Haykin, 2008, for comprehensive mathematical details).

Number of Hidden Neurons

The researcher determines the number of neurons (processing units) that exist within the hidden layer. It should be understood, however, that if an insufficient number of neurons is employed, the network may not be able to identify the intricacies of a potentially complex training data set. This type of problem is often referred to as "underfitting the data." On the other hand, if too many neurons are employed within the

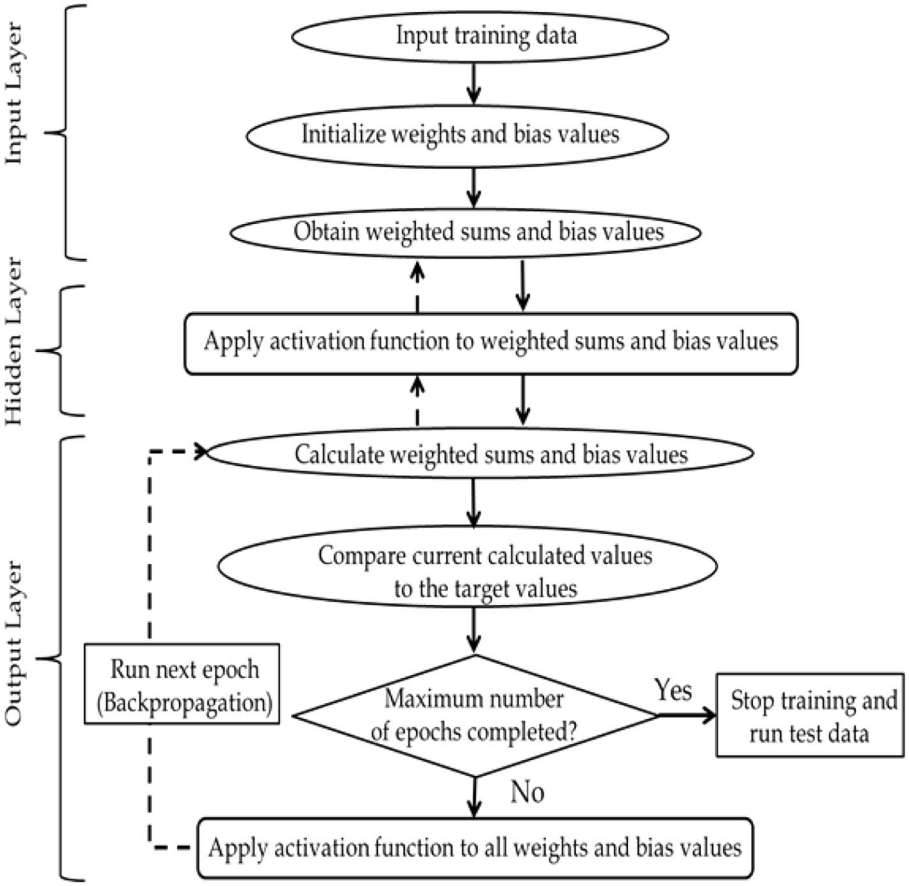


Fig. 15 Illustration of the feedforward backpropagation algorithm. During the forward pass (*solid arrow lines*), input and bias values enter at the top and are passed downward and then recycled during backpropagation (*dashed arrow lines*)

hidden layer, the network attempts to model all of the random noise that usually exists within any complex training data set. In doing so, the idiosyncratic distribution of output values tends to become overly explicit, and the random noise may become incorporated within the network's overall modeling of the training data, resulting in the network's inability to formulate a generalized model of the training data. Such an outcome is frequently described as "overfitting" (see Haykin, 2008, for a discussion of these and other related issues).

Learning Rate and Momentum

As noted within Table 1, the learning rate controls the size and speed of the weight and bias values that are updated during each training epoch. Similar to the learning rate, the momentum modulates the size of the updated values and also helps the network avoid inaccurate updates (often referred to as local minima). Combining high learning rates and momentum terms (e.g., both values at or near 1) increases the likelihood that the

CM may overestimate the best weight and bias values needed to produce an accurate model (see Hagan et al., 2002, for an in-depth discussion).

Software and Input Values

This article illustrates the application of a neural network using the Windows version of C#. The training and test values employed as exemplars within this article were obtained from a study originally conducted by Tovar and Chavez (2012) and replicated by Vernucio and Debert (2016). The feedforward backpropagation algorithm that operates within the EVA application was originally developed and described by McCaffrey (2015; refer to Haykin, 2008, for a related discussion). A functional beta Windows version of EVA (with training and test values) is downloadable for academic researchers from www.chris-ninness.com.

The EVA software is enclosed within a zipped or compressed folder in conjunction with the training data and test data files. The current beta version of EVA runs on most Windows operating systems; however, before running the current Windows version of EVA, some Windows machines may require the installation of a freely available and downloadable Microsoft program (http://go.microsoft.com/fwlink/?LinkID=145727&clcid=0x894). The user must right-click and extract all files prior to beginning the installation process. Subsequent to extraction, the user opens the EVA folder and clicks the EVA icon. If a Microsoft or other antivirus warning appears (and it will), researchers are advised to read the following information prior to installing and running the program.

The current beta version of the EVA application was designed to run on Windows 7–10 operating systems (a Mac OS version is in progress). All versions of the EVA CM neural network system are and will remain freely accessible to interested academic users; however, the current beta version of EVA is designed for academic research and demonstration purposes only. The authors and the journal (*The Behavior Analyst*) assume no liability for any damages associated with using, modifying, or distributing this program. The program has not been extensively field tested with input values (data sets) beyond those described in this article. This program is made available to interested educators or researchers without cost and without any warranties or provision for support. The authors and the journal assume no responsibility or liability for the use of the program and do not provide any certifications, licenses, or titles under any patent, copyright, or government grant. The authors and the journal make no representations or assurances with regard to the security, functionality, or other components of the program. There are unidentifiable hazards associated with installing and running any software application, and users and researchers are responsible for determining the extent to which this program is compatible with the computer and other software currently installed on the user's computer.

References

Abbass, H. A. (2002). An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artificial Intelligence in Medicine, 25*, 265–281. doi:10.1016/s0933-3657(02)00028-3.

Aleven, V. (2013). Help seeking and intelligent tutoring systems: theoretical perspectives and a step towards theoretical integration. In R. Azevedo & V. Aleven (Eds.), *International handbook of metacognition and learning technologies* (pp. 311–335). New York, NY: Springer. doi:10.1007/978-1-4419-5546-3_21.

Allamehzadeh, M., & Mokhtari, M. (2003). Prediction of aftershocks distribution using self-organizing feature maps (SOFM) and its application on the Birjand-Ghaen and Izmit earthquakes. *Journal of Seismology and Earthquake Engineering, 5*, 1–15. doi:10.1016/j.quaint.2012.07.059.

Arciniegas, I., Daniel, B., & Embrechts, M. J. (2001). Exploring financial crises data with self-organizing maps (SOM). In N. Allinson, L. Allinson, H. Yin, & J. Slack (Eds.), *Advances in self-organizing maps* (pp. 30–39). London, England: Springer-Verlag.

Arntzen, E., & Holth, P. (1997). Probability of stimulus equivalence as a function of training design. *The Psychological Record, 47*, 309–320.

Barnes, D., & Hampson, P. J. (1993). Stimulus equivalence and connectionism: implications for behavior analysis and cognitive science. *Psychological Record, 43*, 617–638.

Bullinaria, J. A. (1997). Modeling reading, spelling, and past tense learning with artificial neural networks. *Brain and Language, 59*, 236–266. doi:10.1006/brln.1997.1818.

Burgos, J. E. (2007). Autoshaping and automaintenance: a neural-network approach. *Journal of the Experimental Analysis of Behavior, 88*, 115–130. doi:10.1901/jeab.2007.75-04.

Cohen, A., & Sackrowitz, H. B. (2002). Inference for the model of several treatments and a control. *Journal of Statistical Planning and Inference, 107*, 89–101. doi:10.1016/s0378-3758(02)00245-8.

Connell, J. E., & Witt, J. C. (2004). Applications of computer-based instruction: using specialized software to aid letter-name and letter-sound recognition. *Journal of Applied Behavior Analysis, 37*, 67–71. doi:10.1901/jaba.2004.37-67.

Critchfield, T. S., & Fienup, D. M. (2008). Stimulus equivalence. In S. F. Davis & W. F. Buskist (Eds.), *21st century psychology: a reference handbook* (pp. 360–372). Thousand Oaks, CA: Sage.

Critchfield, T. S., & Fienup, D. M. (2010). Using stimulus equivalence technology to teach about statistical inference in a group setting. *Journal of Applied Behavior Analysis, 43*, 437–462. doi:10.1901/jaba.2010.43-763.

Critchfield, T. S., & Fienup, D. M. (2013). A "happy hour" effect in translational stimulus relations research. *Experimental Analysis of Human Behavior Bulletin, 29*, 2–7.

Cullinan, V., Barnes, D., Hampson, P. J., & Lyddy, F. (1994). A transfer of explicitly and nonexplicitly trained sequence responses through equivalence relations: an experimental demonstration and connectionist model. *The Psychological Record, 44*, 559–585.

De Rose, J. C., De Souza, D. G., & Hanna, E. S. (1996). Teaching reading and spelling: exclusion and stimulus equivalence. *Journal of Applied Behavior Analysis, 29*, 451–469. doi:10.1901/jaba.1996.29-451.

Desmarais, M. C., Meshkinfam, P., & Gagnon, M. (2006). Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction, 16*, 403–434. doi:10.1007/s11257-006-9016-3.

Desmarais, M. C., & Pu, X. (2005). A Bayesian inference adaptive testing framework and its comparison with item response theory. *International Journal of Artificial Intelligence in Education, 15*, 291–323. doi:10.1007/11527886_51.

Donahoe, J. W., & Burgos, J. E. (2000). Behavior analysis and revaluation. *Journal of the Experimental Analysis of Behavior, 74*, 331–346. doi:10.1901/jeab.2000.74-331.

Erdal, H. I., & Ekinci, A. (2013). A comparison of various artificial intelligence methods in the prediction of bank failures. *Computational Economics, 42*, 199–215. doi:10.1007/s10614-012-9332-0.

Feng, M., Heffernan, N. T., & Koedinger, K. R. (2009). Addressing the assessment challenge in an intelligent tutoring system that tutors as it assesses. *User Modeling and User-Adapted Interaction, 19*, 243–266. doi:10.1007/s11257-009-9063-7.

Fienup, D. M., Covey, D. P., & Critchfield, T. S. (2010). Teaching brain–behavior relations economically with stimulus equivalence technology. *Journal of Applied Behavior Analysis, 43*, 19–33. doi:10.1901/jaba.2010.43-19.

Fienup, D. M., & Critchfield, T. S. (2010). Efficiently establishing concepts of inferential statistics and hypothesis decision making through contextually controlled equivalence classes. *Journal of Applied Behavior Analysis, 43*, 19–33. doi:10.1901/jaba.2010.43-437.

Fienup, D. M., & Critchfield, T. S. (2011). Transportability of equivalence-based programmed instruction: efficacy and efficiency in a college classroom. *Journal of Applied Behavior Analysis, 43*, 763–768. doi:10.1901/jaba.2011.44-435.

Fienup, D. M., Critchfield, T. S., & Covey, D. P. (2009). Building contextually-controlled equivalence classes to teach about inferential statistics: a preliminary demonstration. *Experimental Analysis of Human Behavior Bulletin, 27*, 1–10.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition, 28*, 3–71. doi:10.1016/0010-0277(88)90031-5.

Guo, D., Liao, K., & Morgan, M. (2007). Visualizing patterns in a global terrorism incident database. *Environment and Planning B: Planning and Design, 34*, 767–784. doi:10.1068/b3305.

Hagan, M., Demuth, H., & Beale, M. (2002). *Neural network design*. Boston, MA: PWS.

Hamilton, B. E., & Silberberg, A. (1978). Contrast and autoshaping in multiple schedules varying reinforcer rate and duration. *Journal of the Experimental Analysis of Behavior, 30*, 107–122. doi:10.1901/jeab.1978.30-107.

Hayes, S. C., Fox, E., Gifford, E. V., Wilson, K. G., Barnes-Holmes, D., & Healy, O. (2001). Derived relational responding as learned behavior. In S. C. Hayes, D. Barnes-Holmes, & B. Roche (Eds.), *Relational frame theory: a post-Skinnerian account of human language and cognition* (pp. 21–50). New York, NY: Plenum.

Haykin, S. O. (2008). *Neural networks and learning machines* (3rd ed.). Upper Saddle River, NJ: Pearson Education.

Heller, J., Steiner, C., Hockemeyer, C., & Albert, D. (2006). Competence-based knowledge structures for personalised learning. *International Journal on E-Learning, 5*, 75–88.

Huang, Y., Chen, J., Chang, Y., Huang, C., Moon, W. K., Kuo, W., et al. (2013). Diagnosis of solid breast tumors using vessel analysis in three-dimensional power Doppler ultrasound images. *Journal of Digital Imaging, 26*, 731–739. doi:10.1007/s10278-012-9556-5.

Kemp, S. N., & Eckerman, D. A. (2001). Situational descriptions of behavioral procedures: the in situ testbed. *Journal of the Experimental Analysis of Behavior, 75*, 135–164. doi:10.1901/jeab.2001.75-135.

Khan, M. R., & Ondrusek, C. (2000). Short-term electric demand prognosis using artificial neural networks. *Electrical Engineering, 51*, 296–300.

Knutti, R., Stocker, T. F., Joos, F., & Plattner, G. K. (2003). Probabilistic climate change projections using neural networks. *Climate Dynamics, 21*, 257–272. doi:10.1007/s00382-003-0345-1.

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The knowledge-learning-instruction framework: bridging the science-practice chasm to enhance robust student learning. *Cognitive Science, 36*, 757–798. doi:10.1111/j.1551-6709.2012.01245.x.

LeBlanc, L. A., Miguel, C. F., Cummings, A. R., Goldsmith, T. R., & Carr, J. E. (2003). The effects of three stimulus-equivalence testing conditions on emergent US geography relations of children diagnosed with autism. *Behavioral Interventions, 18*, 279–289. doi:10.1002/bin.144.

Lovett, S., Rehfeldt, R. A., Garcia, Y., & Dunning, J. (2011). Comparison of a stimulus equivalence protocol and traditional lecture for teaching single-subject designs. *Journal of Applied Behavior Analysis, 44*, 819–833. doi:10.1901/jaba.2011.44-819.

Lyddy, F., & Barnes-Holmes, D. (2007). Stimulus equivalence as a function of training protocol in a connectionist network. *Journal of Speech and Language Pathology and Applied Behavior Analysis, 2*, 14–24. doi:10.1037/h0100204.

Lyddy, F., Barnes-Holmes, D., & Hampson, P. J. (2001). A transfer of sequence function via equivalence in a connectionist network. *The Psychological Record, 51*, 409–428. doi:10.1037/h0100204.

Maqsood, I., Khan, M. R., & Abraham, A. (2004). An ensemble of neural networks for weather forecasting. *Neural Computing and Applications, 13*, 112–122. doi:10.1007/s00521-004-0413-4.

McCaffrey, J. (2014). Neural networks using C# succinctly [Blog post]. Retrieved from https://jamesmccaffrey.wordpress.com/2014/06/03/neural-networks-using-c-succinctly

McCaffrey, J. (2015). Coding neural network back-propagation using C#. *Visual Studio Magazine.* Retrieved from https://visualstudiomagazine.com/articles/2015/04/01/back-propagation-using-c.aspx

McClelland, J. L., & Rumelhart, D. E. (1986). *Parallel distributed processing, vol. 2: psychological and biological models*. Cambridge, MA: MIT Press.

Nason, S., & Zabrucky, K. (1988). A program for comprehension monitoring of text using HyperCard for the Macintosh. *Behavior Research Methods, Instruments, & Computers, 20*, 499–502.

Ninness, C., Henderson, R., Ninness, C., & Halle, S. (2015). Probability pyramiding revisited: univariate, multivariate, and neural networking analyses of complex data. *Behavior and Social Issues, 24*, 164–186. doi:10.5210/bsi.v24i0.6048.

Ninness, C., Lauter, J., Coffee, M., Clary, L., Kelly, E., Rumph, M., et al. (2012). Behavioral and biological neural network analyses: a common pathway toward pattern recognition and prediction. *The Psychological Record, 62*, 579–598. doi:10.5210/bsi.v22i0.4450.

Ninness, C., Rumph, M., Clary, L., Lawson, D., Lacy, J. T., Halle, S., et al. (2013). Neural network and multivariate analysis: pattern recognition in academic and social research. *Behavior and Social Issues, 22*, 49–63. doi:10.5210/bsi.v22i0.4450.

Ninness, C., Rumph, R., McCuller, G., Harrison, C., Vasquez, E., Ford, A., et al. (2005). A relational frame and artificial neural network approach to computer-interactive mathematics. *The Psychological Record, 55*, 561–570. doi:10.1007/bf03395503.

Oğcu, G., Demirel, O. F., & Zaim, S. (2012). Forecasting electrical consumption with neural networks and support vector regression. *Procedia – Social and Behavioral Sciences, 58*, 1576–1585. doi:10.1016/j.sbspro.2012.09.1144.

Rumelhart, D. E., Hinton, G. E., & Williams, D. C. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536. doi:10.1038/323533a0.

Sidman, M., & Cresson, O. (1973). Reading and crossmodal transfer of stimulus equivalences in severe retardation. *American Journal of Mental Deficiency, 77*, 515–523.

Sidman, M., & Tailby, W. (1982). Conditional discrimination vs. matching to sample: an expansion of the testing paradigm. *Journal of the Experimental Analysis of Behavior, 37*, 5–22.

Steele, D. M., & Hayes, S. C. (1991). Stimulus equivalence and arbitrarily applicable relational responding. *Journal of the Experimental Analysis of Behavior, 56*, 519–555. doi:10.1901/jeab.1991.56-519.

Stromer, R., Mackay, H. A., & Stoddard, L. T. (1992). Classroom applications of stimulus equivalence technology. *Journal of Behavioral Education, 2*, 225–256. doi:10.1007/bf00948817.

Tovar, A. E., & Chavez, A. T. (2012). A connectionist model of stimulus class formation with a yes/no procedure and compound stimuli. *The Psychological Record, 62*, 747–762. doi:10.1007/s40732-016-0184-1.

Vernucio, R. R., & Debert, P. (2016). Computational simulation of equivalence class formation using the go/no-go procedure with compound stimuli. *The Psychological Record, 66*, 439–449. doi:10.1007/s40732-016-0184-1.

Walker, D., Rehfeldt, R. A., & Ninness, C. (2010). Using the stimulus equivalence paradigm to teach course material in an undergraduate rehabilitation course. *Journal of Applied Behavior Analysis, 43*, 615–633. doi:10.1901/jaba.2010.43-615.

Wolberg, W. (1992). *Breast cancer Wisconsin (diagnostic) data set* [UCI Machine Learning Repository]. Retrieved from http://archive.ics.uci.edu/ml/

You, H., & Rumbe, G. (2010). Comparative study of classification techniques on breast cancer FNA biopsy data. *International Journal of Artificial Intelligence and Interactive Multimedia, 3*, 5–12. doi:10.9781/ijimai.2010.131.