UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE PSICOLOGIA

RENATO ROBERTO VERNUCIO

Simulação computacional do comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos

São Paulo

2015

RENATO ROBERTO VERNUCIO

Simulação computacional do comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos

Dissertação apresentada ao Instituto de Psicologia da Universidade de São Paulo como parte dos requisitos para obtenção do título de Mestre em Psicologia.

Área de concentração:
Psicologia Experimental

Orientadora:
Prof.ª Dr.ª Paula Debert

São Paulo

2015

Catalogação na publicação
Biblioteca Dante Moreira Leite
Instituto de Psicologia da Universidade de São Paulo

Vernucio, Renato Roberto.

Simulação computacional do comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos / Renato Roberto Vernucio; orientadora Paula Debert. -- São Paulo, 2015.

69 f.

Dissertação (Mestrado – Programa de Pós-Graduação em Psicologia. Área de Concentração: Psicologia Experimental) – Instituto de Psicologia da Universidade de São Paulo.

1. Simulação 2. Inteligência artificial 3. Equivalência de estímulos 4. Estímulo composto 5. Procedimento *go/no-go* I. Título.

BF186

Nome: Vernucio, Renato Roberto

Título: Simulação computacional do comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos

Dissertação apresentada ao Instituto de Psicologia da Universidade de São Paulo para obtenção do título de Mestre em Psicologia

Aprovado em:

Banca examinadora

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Prof. Dr. _____

Instituição: _____ Assinatura: _____

RESUMO

Vernucio, R. R. (2015). Simulação computacional do comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos. Dissertação de mestrado, Instituto de Psicologia, Universidade de São Paulo, São Paulo.

A partir do treino de relações condicionais podem emergir relações que não foram diretamente treinadas, formando-se classes de equivalência. As pesquisas sobre equivalência de estímulos comumente adotaram o procedimento *matching-to-sample* (MTS) e utilizam humanos como sujeitos experimentais. Mais recentemente, alguns estudos utilizaram modelos computacionais para simular o comportamento de formação de classes de equivalência. O modelo computacional comumente utilizado chama-se RELNET, que simula tentativas de treino e teste de relações condicionais usando-se o MTS. A diferenciação entre estímulos modelo e de comparação, indispensável no MTS, implica em dificuldades operacionais para simulações computacionais. Por isso, novos estudos buscaram simular o comportamento de formação de classes de equivalência utilizando-se procedimentos alternativos ao MTS, que não diferenciam funções em específico aos estímulos antecedentes. O presente trabalho avaliou a possibilidade de desenvolver um novo modelo computacional para simular o comportamento de formação de classes de equivalência utilizando o procedimento *go/no-go* com estímulos compostos. Foi utilizada Linguagem C e FANN para a programação das simulações. Para assegurar que o uso dessas ferramentas não implicaria em mudanças nos resultados, no Experimento 1, foi feita uma replicação sistemática de Tovar e Torres (2012), que investigou a possibilidade de simular o comportamento de formação de classes de equivalência pelo procedimento *yes-no*, alterando-se apenas a utilização da Linguagem C e FANN. Como resultados, cinco das seis execuções apresentaram formação de classes de equivalência, conforme obtido no trabalho de Tovar e Torres (2012), indicando que a utilização de Linguagem C e FANN não implicou em alterações nos resultados. O Experimento 2 teve como objetivo avaliar a possibilidade de simular o comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos, baseando-se no método de Tovar e Torres (2012). Da mesma forma que em Tovar e Torres (2012), foi feito um treino de classe adicional, que seria análogo à simulação do conhecimento pré-experimental de participantes humanos com capacidades linguísticas. Como resultados, obteve-se que quatro de seis execuções apresentaram formação de classes de equivalência, indicando que foi possível simular o comportamento de formação de classes de equivalência usando o procedimento *go/no-go* com estímulos compostos. Para avaliar se o treino de classe adicional é indispensável em simulações envolvendo o procedimento *go/no-go*, o Experimento 3 usou os mesmos parâmetros do Experimento 2 sem o treino de classe adicional. Como resultados, duas das seis execuções apresentaram formação de classes de equivalência. Isso indica que para simular o comportamento de formação de classes de equivalência usando o procedimento *go/no-go* com estímulos compostos também é preciso haver treino de classe adicional. Dada a importância desse treino, no Experimento 4 foi avaliado se a partir do treino de uma classe adicional contendo três elementos seria possível simular a formação de classes de equivalência contendo quatro elementos. Como resultados, quatro das seis execuções apresentaram formação de classes de equivalência. Em conjunto, os resultados mostram que o modelo proposto é capaz de simular o comportamento de formação de classes de equivalência, sem diferenciar as funções de estímulo modelo e de comparação, sendo uma alternativa ao RELNET.

Palavras-chave: simulação, inteligência artificial, equivalência de estímulos, estímulo composto, *go/no-go*.

ABSTRACT

Vernucio, R. R. (2015). Computer simulation of equivalence class formation using the go/no-go procedure with compound stimuli. Dissertação de mestrado, Instituto de Psicologia, Universidade de São Paulo, São Paulo.

Relations that were not directly trained can emerge from conditional discrimination training that establishes equivalence classes. Equivalence classes are usually established using the matching-to-sample procedure (MTS) with humans as experimental subjects. More recently, some studies used computational models to simulate equivalence class formation. The computational model that is commonly used is named RELNET and it simulates training and testing conditional relations using MTS procedure. The distinction between sample stimulus function and comparison stimuli function found in the MTS procedure causes operational difficulties in computational simulations. For this reason, new studies tried to simulate the establishment of equivalence classes using alternative procedures to the MTS that don't specify sample and comparison functions. The present study assessed the possibility to develop a new computational model to simulate equivalence class formation using the go/no-go procedure with compound stimuli. C Language and FANN were used to program the simulations. To ensure that the use of these tools wouldn't change results, in Experiment 1, a systematic replication of Tovar and Torres (2012) was conducted to evaluate the possibility of simulating equivalence class formation with the yes-no procedure using C Language and FANN. Five of the six runs exhibited equivalence class formation indicating that the use of C Language and FANN produced the same results as was obtained by Tovar and Torres (2012). Experiment 2 evaluated the possibility of simulating equivalence class formation using go/no-go procedure with compound stimuli. Additional class training was provided as in Tovar and Torres (2012) to simulate human's pre-experimental history. Four of the six runs exhibited equivalence class formation. Therefore, it was possible to simulate equivalence class formation using go/no-go procedure with compound stimuli. To evaluate if the additional class training was necessary to simulate the establishment of equivalence classes with the go/no-go procedure, Experiment 3 used the same parameters as in the Experiment 2 without the additional class training. Two of the six runs exhibited equivalence class formation, indicating that the additional class training is necessary to simulate equivalence class formation also in simulations with the go/no-go procedure with compound stimuli. In Experiment 4, it was evaluated if training one additional class with three elements would be sufficient to simulate the establishment of equivalence classes with four elements. Four of the six runs exhibited the establishment of equivalence classes. The results of the four experiments show that the simulation model investigated can simulate equivalence class formation using the go/no-go procedure and it is an alterative to RELNET.

Keywords: simulation, artificial intelligence, stimuli equivalence, compound stimulus, go/no-go.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

De acordo com Sidman e Tailby (1982), a partir do treino de relações condicionais, podem emergir novas relações que não foram diretamente treinadas. Tais relações entre estímulos possuem propriedades específicas denominadas reflexividade, simetria, transitividade e equivalência. Assim, estímulos podem ser considerados equivalentes quando é evidenciado que estabelecem entre si todas essas relações.

Um procedimento comumente utilizado para o estabelecimento de relações de equivalência é o *matching-to-sample* (MTS). O procedimento MTS padrão consiste na apresentação de um estímulo condicional, chamado estímulo modelo, e de dois ou mais estímulos de escolha, chamados estímulos de comparação. Os estímulos de comparação que são relacionados a cada estímulo modelo são determinados pelo experimentador. A resposta de escolher o estímulo de comparação determinado como correspondente ao estímulo modelo a cada tentativa é seguida de reforço, enquanto a resposta de escolher os demais estímulos não é seguida de consequências programadas. Os diferentes estímulos são comumente representados esquematicamente por uma letra e um número, sendo que o número representa a classe de estímulos. Assim, por exemplo, diante do estímulo modelo A1, com a apresentação dos estímulos de comparação B1 e B2, a escolha de B1 é seguida de reforço enquanto a escolha de B2 não é seguida de consequências programadas. Já na presença do estimulo modelo A2, com a apresentação dos estímulos de comparação B1 e B2, a escolha de B2 é seguida de reforço enquanto a escolha de B1 não é seguida de consequências programadas. Dessa forma, são estabelecidas relações condicionais entre A1 e B1 e entre A2 e B2. Para estabelecer classes de estímulos equivalentes, devem ser treinadas, da mesma maneira, as relações BC. Seguindo o exemplo citado, a escolha de C1 diante do estímulo modelo B1 é seguida de reforço, bem como a escolha de C2 diante de B2. Já a escolha de C1 diante de B2, bem como a escolha de C2 diante de B1 não é seguida de consequências programadas. Após esses treinos, são conduzidos testes das relações emergentes de simetria, transitividade e equivalência que atestariam a formação de classes de equivalência (Sidman & Tailby, 1982).

Uma relação de reflexividade pode ser testada quando, em extinção, o sujeito escolhe o estímulo de comparação igual ao estímulo modelo, por exemplo, escolhendo A1 entre outros estímulos de comparação quando o próprio A1 é o estímulo modelo. Uma relação de simetria é verificada, por exemplo, quando após o treino AB é constatada, em extinção, a relação BA. A relação de transitividade é constatada quando emerge uma relação que não foi diretamente treinada contendo distância nodal de pelo menos um elemento. Assim, por

exemplo, após os treinos AB e BC, verifica-se transitividade quando, em extinção, o participante demonstra a relação AC. Por fim, testa-se a relação de equivalência verificando-se a relação simétrica de uma relação de transitividade, como, por exemplo, após a emergência da relação de transitividade AC, constatando-se, em extinção, a relação CA. A partir desses treinos e testes é possível estudar experimentalmente como estímulos relacionam-se e podem ser substituídos entre si (Sidman, 1994).

As pesquisas sobre equivalência de estímulos comumente adotaram o procedimento MTS para treino e teste de relações condicionais utilizando participantes humanos (Sidman, 1994). Mais recentemente, foram desenvolvidas pesquisas que utilizaram modelos computacionais para simular o comportamento de formação de classes de equivalência, sendo capazes de reproduzir desempenhos observados em humanos (e.g., Barnes & Hampson, 1993; Cullinan, Barnes, Hampson & Lyddy, 1994; Lyddy & Barnes- Holmes, 2007; Lyddy, Barnes-Holmes, & Hampson, 2001; Okada, Sakagami & Yamakawa, 2005; Tovar & Torres, 2012). O tipo de modelo computacional utilizado pelos estudos é denominado de rede neural artificial[1] (RNA), rede neural, rede conexionista ou apenas rede (Haykin, 2001).

Simulações computacionais de comportamento, de maneira geral, têm a vantagem de oferecer um maior controle das variáveis independentes, garantindo um ambiente estável e com grau de controlabilidade difícil de ser atingido em experimentos com humanos e outros animais. Variáveis como aprendizagem pré-experimental podem ser facilmente controladas e manipuladas (Lyddy & Barnes-Holmes, 2007). Por meio de modelos computacionais é possível fazer simulações que envolvem grande escala de tempo, como, por exemplo, simular evoluções sociais ao longo de séculos ou milênios (McElreath & Boyd, 2008). Os modelos computacionais podem simular comportamentos de forma a ir além das limitações dos desempenhos de organismos vivos, como, por exemplo, desenvolver uma ferramenta capaz de simular centenas de milhares de variações de respostas em um contexto de discriminação simples em diferentes esquemas de reforço (Kemp & Eckerman, 2001), ininterruptamente, sem se preocupar com fadiga ou outras variáveis indesejáveis. Impedimentos provenientes de questões éticas da experimentação com humanos ou outros animais não acontecem em pesquisas com simulações computacionais, dado que não há seres vivos envolvidos e os experimentos são feitos no que se denomina *in silico*. A partir de simulações computacionais de comportamento é possível fazer previsões e gerar conhecimentos novos (McClelland, 2009). A infra-estrutura exigida para realizar experimentos envolvendo simulações

computacionais resume-se a um ou mais computadores e seu custo de manutenção é menor do que o custo de manutenção de um laboratório de experimentação animal ou humana. Além disso, simulações computacionais podem ser extremamente rápidas, após serem apropriadamente programadas.

Lyddy e Barnes-Holmes (2007) utilizaram simulações computacionais para avaliar comparativamente os desempenhos de formação de classes de equivalência em treinos linear e *one-to-many* usando o procedimento MTS. A simulação foi feita usando-se o *Tlearn* (Plunkett & Elman, 1997), um software genérico para programação de RNAs. Foram simuladas as tentativas de treino e teste em que eram exibidos um estímulo modelo, três estímulos de comparação e era gerada uma resposta de escolha. Os estímulos presentes e a resposta de escolha a cada tentativa eram representados computacionalmente por um recurso das RNAs denominado arquitetura de rede. Na Figura 1, está esquematizada a arquitetura de rede utilizada em Lyddy e Barnes-Holmes (2007).

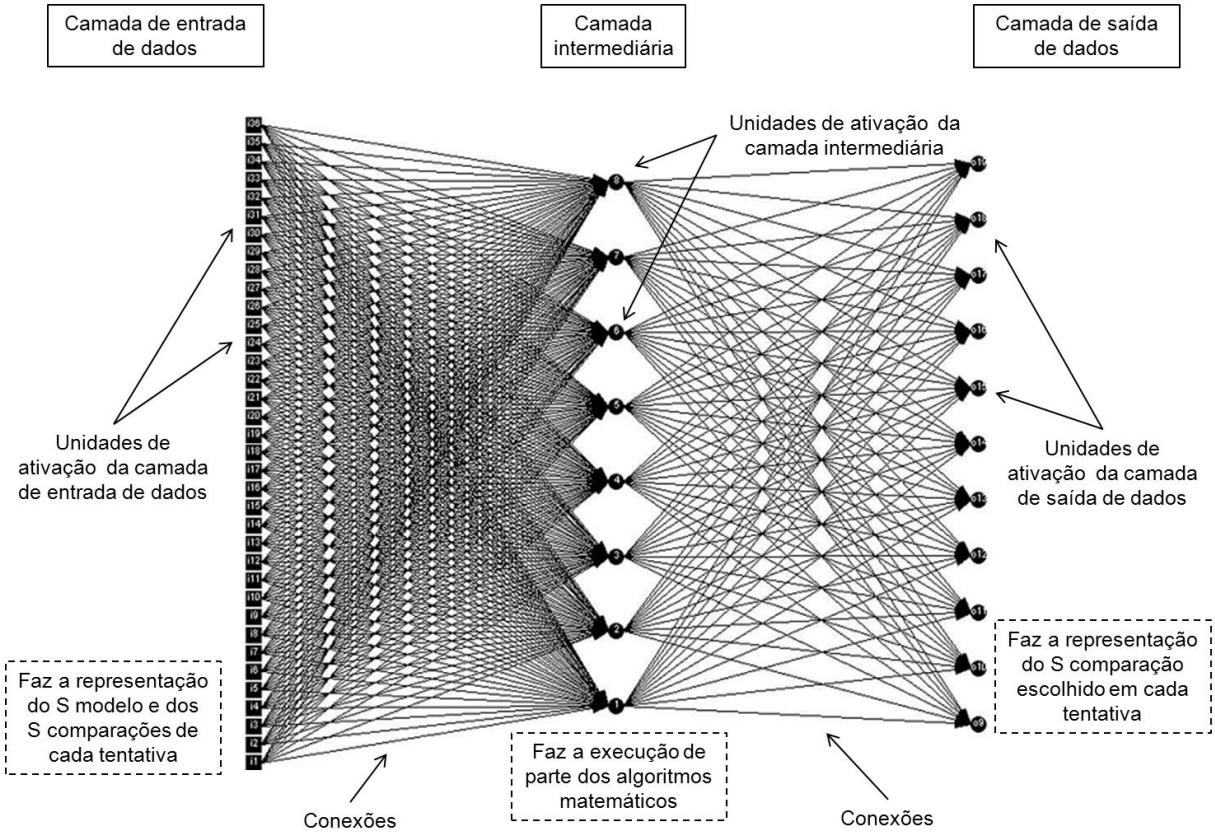

Figura 1. Arquitetura de rede utilizada em Lyddy e Barnes-Holmes (2007), tendo sido o texto acrescentado.

A arquitetura representada na Figura 1 é composta por partes de códigos de programação denominados unidades de ativação, os quais estão organizados em agrupamentos chamados de camadas. A arquitetura apresentada é composta por três camadas. Do lado esquerdo da figura está a camada de entrada de dados, responsável por fazer a representação computacional do estímulo modelo e dos três estímulos de comparação exibidos a cada tentativa. Essa representação era feita por meio da atribuição de valores em específico para cada uma das 36 unidades da qual essa camada era composta. O conjunto de valores da camada de entrada de dados é denominado vetor de entrada. As linhas da figura representam conexões entre unidades de ativação. Cada conexão indica que o valor de saída gerado por uma unidade à esquerda é passado como valor de entrada para a unidade à direita, multiplicado por uma variável denominada peso da conexão, ou apenas peso. Cada conexão tem um peso distinto. Denomina-se *feed-forward* quando as unidades são dispostas unidirecionalmente, sendo os valores passados da esquerda para a direita. Assim, os valores da camada de entrada, multiplicados pelos pesos de suas conexões, eram passados para as unidades da camada intermediária, representada ao centro da figura. Nessa camada, os valores eram modificados por algoritmos matemáticos característicos da arquitetura *feed-forward* (ver Haykin, 2001), e passados para a a última camada, representada do lado direito da figura. A camada de saída de dados fazia a representação da resposta de escolha. Essa representação era feita por meio de valores em específico gerados por cada uma das 11 unidades da qual essa camada era composta. O conjunto de valores da camada de saída de dados é denominado vetor de saída. Em resumo, a partir de um vetor de entrada, que corresponderia à representação da presença de um estímulo modelo e três estímulos de comparação em específico na tentativa, após o processamento dos algoritmos matemáticos, era gerado um vetor de saída, que corresponderia à representação da resposta de escolha por um estímulo de comparação naquela tentativa.

Lyddy e Barnes-Holmes (2007) argumentam que, em experimentos com humanos, sujeitos com habilidades linguísticas já possuem repertórios prévios de formação de classes de equivalência, como também classes de equivalência previamente formadas anteriormente ao experimento. Sendo assim, simulações computacionais de desempenhos humanos devem também estabelecer classes de equivalência previamente ao que seria o treino propriamente dito. Por isso, foram utilizados dois conjuntos de estímulos (XYZ e ABC). Com o primeiro conjunto de estímulos foi simulado o conhecimento pré-experimental, realizando-se um pré-treino. O segundo conjunto de estímulos foi utilizado para realizar o treino que avaliou os protocolos de treino linear e *one-to-many*.

No pré-treino, as relações XY, YZ e XZ foram diretamente treinadas para três classes de estímulos. Assim, por exemplo, para o treino da relação XY, a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo X1 e dos estímulos de comparação Y1, Y2 e Y3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação Y1. Já a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo X2 e dos estímulos de comparação Y1, Y2 e Y3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação Y2. E a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo X3 e dos estímulos de comparação Y1, Y2 e Y3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação Y3.

A partir das relações que deveriam ser estabelecidas durante o pré-treino, foi criado um *dataset*, que é o conjunto de vetores de entrada e seus vetores de saída correspondentes esperados (por exemplo, vetor de entrada que representava a exibição do estímulo modelo X1 e dos estímulos de comparação Y1, Y2 e Y3, e seu vetor de saída correspondente, que representava a escolha de Y1). Contudo, inicialmente os vetores de saída resultantes não correspondiam aos vetores de saída esperados, conforme o *dataset*. Quando a escolha era incorreta, isto é, quando os valores do vetor de saída não correspondiam perfeitamente aos valores esperados para aquela tentativa, um procedimento de correção era realizado utilizando-se o algoritmo de aprendizagem computacional *backward propagation* (Rumelhart, Hinton & Williams, 1986). Esse algoritmo modificava os valores dos pesos das conexões entre as unidades de ativação, de forma que fossem gerados novos vetores de saída com valores mais próximos do desejado, conforme o *dataset*. No início da simulação, os pesos iniciais foram atribuídos aleatoriamente com valores entre 0 e 1. Ao longo das tentativas, os pesos das conexões eram alterados pelo algoritmo *backward propagation*, de maneira que gradativamente iam assumindo valores otimizados, que produziam vetores de saída cada vez mais próximos dos valores esperados para todas as relações treinadas. O procedimento era repetido até que um critério de parada fosse atingido, usando-se a medida de Erro Quadrático Médio (EQM) (ver Bussab & Morettin, 2010). O EQM, resumidamente, avaliava quão discrepantes os vetores de saída gerados estavam em relação aos vetores de saída esperados. Quando o critério EQM $\leq 0{,}05$ foi atingido, considerou-se que todos os vetores de entrada geravam vetores de saída suficientemente próximos ao esperado, conforme o *dataset*. Em

outras palavras, para fins de simulação, os estímulos de comparação estavam sendo escolhidos conforme o esperado de acordo com o estímulo modelo apresentado em cada tentativa. Portanto, ao final do pré-treino, era possível considerar que as relações XY, YZ e XZ haviam sido estabelecidas para três conjuntos de estímulos. A execução do *backward propagation* consistiu em um procedimento computacional análogo ao envolvido na aprendizagem de relações condicionais em experimentos com seres vivos.

Após o pré-treino, foi realizado o treino com o segundo conjunto de estímulos (ABC). Para o protocolo linear, foram treinadas as relações AB e BC, e foi testada a emergência da relação CA. Para o protocolo *one-to-many*, foram treinadas as relações AB e AC, e foi testada a emergência das relações BC e CB. Os treinos utilizando-se o conjunto de estímulos ABC foram realizados da mesma forma como foi feito com o outro conjunto de estímulos. Assim, por exemplo, para o treino da relação AB, a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo A1 e dos estímulos de comparação B1, B2 e B3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação B1. Já a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo A2 e dos estímulos de comparação B1, B2 e B3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação B2. E a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo A3 e dos estímulos de comparação B1, B2 e B3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação B3.

Foi estabelecido um *dataset* para cada protocolo avaliado, contendo os vetores de entrada e seus vetores de saída correspondentes para as relações treinadas (e.g., vetor de entrada que representava a exibição do estímulo modelo A1 e dos estímulos de comparação B1, B2 e B3, e seu vetor de saída correspondente, que representava a escolha de B1). O algoritmo *backward propagation* foi utilizado para alterar os pesos das conexões entre as unidades de ativação até que os vetores de saída gerados fossem próximos dos vetores de saída esperados, atingindo-se o critério de mudança de fase. Para o conjunto de execuções usados para avaliar o protocolo linear, ao final da fase de treino era possível considerar que as relações AB e BC haviam sido estabelecidas. Para o conjunto de execuções usados para avaliar o protocolo *one-to-many*, ao final da fase de treino, era possível considerar que as relações AB e AC haviam sido estabelecidas.

Na fase de teste, não foram utilizados procedimentos de correção. O algoritmo *backward propagation* foi desativado e os pesos das conexões mantiveram-se constantes. Foram então apresentados vetores de entrada diferentes daqueles usados durante a fase de treino e foi avaliado se os vetores de saída gerados indicariam emergência de relações que não haviam sido diretamente treinadas. Para cada protocolo foi testada a relação emergente derivada de cada treino. Assim, por exemplo, para avaliar a emergência da relação CA no protocolo linear, a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo C1 e dos estímulos de comparação A1, A2 e A3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação A1. Já a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo C2 e dos estímulos de comparação A1, A2 e A3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação A2. E a partir do vetor de entrada que fazia a representação computacional da exibição do estímulo modelo C3 e dos estímulos de comparação A1, A2 e A3, era esperado que o vetor de saída resultante na camada de saída de dados correspondesse à representação computacional da escolha do estímulo de comparação A3. Como as relações testadas não foram especificadas no *dataset* e, por isso, não foram diretamente treinadas, desempenhos conforme o esperado indicariam a emergência dessas relações.

Como resultados, obteve-se que as execuções que passaram por relações de treino de acordo com o protocolo linear precisaram de aproximadamente o dobro de tentativas para atingir o critério de mudança de fase durante o treino e, na fase de teste, não apresentaram desempenhos consistentes com a emergência da relação CA, indicando que não houve formação de classes de equivalência. Já as execuções que passaram por relações de treino de acordo com o protocolo *one-to-many* atingiram mais rapidamente o critério de mudança de fase durante o treino e, na fase de teste, apresentaram desempenhos consistentes com a emergência das relações BC e CB, indicando a formação de três classes de equivalência (A1B1C1, A2B2C2, A3B3C3). Em conjunto, os dados sugerem que o protocolo *one-to-many* é mais eficaz que o protocolo linear para treino de relações condicionais e estabelecimento de classes de equivalência. Os resultados obtidos usando-se simulações computacionais foram semelhantes ao que foi encontrado em experimentos envolvendo humanos (e.g., Arntzen & Holth, 1997).

O estudo de Lyddy e Barnes-Holmes (2007) utilizou um modelo computacional derivado do *Network for Relational Responding* (RELNET), proposto inicialmente por Barnes

e Hampson (1993). O RELNET foi originalmente desenvolvido para simular o Experimento 2 de Steele e Hayes (1991), que envolveu controle condicional de segunda ordem e relações relacionadas à Teoria dos Quadros Relacionais (Hayes & Hayes, 1989). A adaptação feita por Lyddy e Barnes-Holmes (2007) manteve os algoritmos matemáticos do RELNET e adequou a arquitetura de rede para simular tentativas de treino e teste envolvendo controle condicional de primeira ordem (i.e., apresentação de um estímulo modelo e três estímulos de comparação) e relações de equivalência.

O RELNET originalmente consistia de uma RNA contendo três camadas: uma camada de entrada de dados, uma camada intermediária e uma camada de saída de dados. A camada de entrada de dados compreendia às representações computacionais do estímulo modelo, dos estímulos de comparação e do estímulo contextual exibidos a cada tentativa. Para a representação dessa informação, o vetor de entrada era subdividido em três elementos (três partes). O primeiro elemento fazia a representação do estímulo modelo e dos estímulos de comparação da tentativa sem especificar qual dos estímulos era o modelo e quais eram comparações. O segundo elemento especificava qual dentre os estímulos apresentados no primeiro elemento era o estímulo modelo. Isso era feito usando-se um recurso computacional que os autores denominaram duplicador do marcador do modelo, o qual fazia a duplicação da representação do estímulo modelo nesse segundo elemento do vetor de entrada. O terceiro elemento fazia a representação do estímulo contextual. A camada intermediária estava conectada com as demais em uma estrutura *feed-forward* (semelhante à arquitetura da Figura 1) e fazia parte dos algoritmos matemáticos. A camada de saída de dados fazia a representação do estímulo de comparação escolhido na tentativa.

A arquitetura e os algoritmos utilizados no RELNET foram tomados como base para o desenvolvimento de outros estudos envolvendo simulações do comportamento de formação de classes de equivalência usando-se o procedimento MTS para treino e teste de relações (e.g., Cullinan, Barnes, Hampson & Lyddy, 1994; Lyddy & Barnes-Holmes, 2007; Lyddy, Barnes-Holmes, & Hampson, 2001). Todos os modelos derivados do RELNET utilizaram em sua arquitetura o duplicador do marcador do modelo para diferenciar as funções de estímulo modelo e de estímulos de comparação.

Tovar e Torres (2012) apontaram que RNAs derivadas do RELNET possuem um problema decorrente da diferenciação entre as funções de estímulo modelo e de estímulos de comparação, feita pelo duplicador do marcador do modelo (segundo elemento do vetor de entrada). Em diferentes tentativas, as unidades de ativação referentes ao duplicador do marcador do modelo permanecem constantes e determinam os valores do vetor de saída.

Sendo assim, os resultados apresentados na fase de teste não são necessariamente determinados pela informação sobre quais são os estímulos modelo e comparação em cada tentativa, mas sim pela informação sobre qual é o estímulo modelo (segundo elemento do vetor de entrada). Segundo Tovar e Torres (2012), o fato de que algumas respostas sejam devidas a informações sobre a marcação do modelo, os resultados apresentados na fase de teste não podem ser considerados emergentes:

> Durante os testes de emergência de relações, os padrões de simulação são idênticos com relação ao segundo elemento da camada de entrada (duplicador do marcador do modelo) para algumas das tentativas (…) sendo assim, mesmo quando o primeiro elemento muda de acordo com a tentativa em particular, o elemento referente ao duplicador do marcador do modelo permanece constante. Isso tem uma influência determinante nos valores de ativação obtidos na camada de saída de dados; em consequência, modelos RELNET não oferecem uma resposta 'emergente' nas tentativas dos testes. As saídas geradas por essas redes resultam de informações previamente aprendidas sobre a marcação do modelo (Tovar & Torres, p. 748, 2012). (grifo acrescentado)

O problema operacional das redes derivadas do RELNET, portanto, provém da necessidade de se diferenciar no vetor de entrada qual é o estímulo modelo e qual é o estímulo de comparação, característica que é indispensável no procedimento MTS. Por esse motivo, Tovar e Torres (2012) apontaram como possível solução que fosse desenvolvido um novo modelo computacional que simulasse o comportamento de formação de classes de equivalência usando-se algum procedimento alternativo ao MTS, que não demandasse a diferenciação das funções de estímulo modelo e de estímulo de comparação:

> Para evitar problemas relacionados à marcação dos estímulos, uma alternativa é projetar uma simulação que não precisará de uma representação para as funções de estímulo modelo e estímulo de comparação. Nesse contexto, os procedimentos *go/no-go* e *yes-no* com estímulos compostos foram sugeridos como alternativas para o procedimento MTS para o estudo de formação de classes de estímulos (Tovar & Torres, 2012, p. 748).

Um dos procedimentos alternativos ao MTS para o estabelecimento de classes de equivalência que os autores sugerem é o *yes-no* (e.g., Fields, Reeve, Varelas, Rosen & Belanich, 1997; Fields, Doran & Marroquin, 2009).

O estudo de Fields et al. (1997) fez uso do procedimento *yes-no* para treino e teste de relações condicionais em humanos. Dezoito adultos participaram do experimento. Na tela de um computador, a cada tentativa eras exibidos dois estímulos sucessivamente, sendo que o

segundo estímulo aparecia após o término da apresentação do primeiro estímulo. Como nesse procedimento eram exibidos compostos de estímulos, as funções de estímulo modelo e estímulo de comparação não eram diferenciadas pelo experimentador. Durante a fase de treino, responder "sim" diante de dos dois estimulos da mesma classe, bem como responder "não" diante de dois estimulos de classes diferentes, era seguida de reforço. Na situação inversa, ou seja, responder "sim" diante de dois estimulos de classes diferentes, ou responder "não" diante de dois estímulos da mesma classe, era apresentada a palavra "errado" na tela. Na fase de treino, foram estabelecidas as relações AB, BC e CD para duas classes de estímulos. Após atingido o critério de mudança de fase, foram testadas, em extinção, relações de simetria (BA, CB e DC), transitividade (AC, BD e AD) e equivalência (CA, DB e DA). Como resultados, obteve-se que 10 dos 18 participantes demonstraram formação de classes de equivalência. Foi encontrado que participantes que não formaram classes de equivalência muitas vezes apresentavam a resposta "não" diante de todas as tentativas de teste. Os autores levantaram a hipótese de que a resposta "não" ficou sob controle da apresentação de combinações de estímulos que não fizeram parte do treino.

Fields et al. (2009) investigaram fatores que poderiam aumentar ou não a chance de que classes de equivalência fossem formadas usando-se o procedimento *yes-no*. No Experimento 1, foi avaliado se os rótulos "sim" e "não" nas teclas que os participantes poderiam apertar produziria resultados diferentes se fossem substituídos pelos rótulos "igual" e "diferente" nas mesmas teclas. Foram treinadas e testadas as mesmas relações do estudo de Fields et al. (1997). Um grupo de participantes foi submetido às etapas de treino e teste com os rótulos "sim" e "não", enquanto um segundo grupo de participantes foi submetido aos mesmos treinos e testes com os rótulos "igual" e "diferente". Como resultados, não foi obtido diferença estatisticamente significativa entre os grupos em relação à quantidadade de participantes que formaram classes de equivalência. Também foi encontrado que a maioria dos participantes que não formaram classes apresentaram a resposta "não" para todos os compostos em testes de transitividade, replicando os resultados de Fields et al. (1997).

No Experimento 2, foi investigado se a probabilidade da formação de classes de equivalência pelo procedimento *yes-no* com os rótulos "sim" e "não" poderia ser aumentada realizando-se um pré-treino, que consistiu na utilização de conjuntos adicionais de estímulos para o estabelecimento de relações lineares e transitivas. Esses conjuntos adicionais de estímulos fizeram parte apenas do pré-treino. Em seguida, foram treinadas e testadas as mesmas relações do Experimento 1. Como resultados, obteve-se que todos os participantes do grupo que passou pelo pré-treino formaram classes de equivalência. No grupo que não passou

pelo pré-treino apenas, aproximadamente, metade dos participantes formaram classes de equivalência, sendo que os participantes que não formaram classes apresentaram o padrão de responder "não" em todas as tentativas do teste de transitividade.

Tomando como base o procedimento *yes-no*, Tovar e Torres (2012) avaliaram a possibilidade de desenvolver um modelo computacional para simular as tentativas de treino e teste de relações condicionais usando-se esse procedimento. Considerando que no *yes-no* não são diferenciadas as funções de estímulo modelo e de estímulo de comparação, a arquitetura de rede do modelo não precisaria fazer uso de um recurso computacional semelhante ao duplicador do marcador do modelo das redes derivadas do RELNET. O estudo foi composto por dois experimentos, sendo o primeiro realizado com humanos e o segundo com simulações computacionais. O primeiro experimento teve como objetivo obter dados de desempenhos humanos para servirem de comparação para os resultados do segundo experimento.

Participaram do primeiro experimento seis adultos com idades entre 19 e 22 anos. O procedimento foi semelhante ao usado no Experimento 1 de Fields et al. (2009) para o grupo cujos rótulos das teclas eram "sim" e "não", com exceção da apresentação dos estímulos, das relações treinadas e testadas, e das consequências às respostas dos participantes. Na tela de um computador eram exibidos dois estímulos simultaneamente a cada tentativa e, usando um mouse, o participante poderia clicar em um botão escrito "sim" ou em um botão escrito "não". A resposta de clicar em "sim" diante de dois estímulos de mesma classe era seguida de distribuição de pontos e de som indicativo de acerto, enquanto a resposta de clicar no botão "não" era seguida de som indicativo de erro. Já diante de dois estímulos de classes diferentes ocorria o inverso, sendo que a resposta de clicar no botão "não" era seguida de distribuição de pontos e de som indicativo de acerto, enquanto a resposta de clicar no botão "sim" era seguida de som indicativo de erro. Foram treinadas as relações AB e BC para dois conjuntos de estímulos. Após atingido o critério de mudança de fase, foi testada, em extinção, a emergência de relações de simetria (BA, CB), transitividade (AC) e equivalência (CA). Como resultados, obteve-se que quatro dos seis participantes apresentaram desempenhos consistentes com a formação de duas classes de equivalência. Observou-se que os participantes que não formaram classes de equivalência apresentaram desempenho semelhante ao encontrado por Fields et al. (2009) nos grupos que não passaram por pré-treino, apresentando altas porcentagens de acertos em testes de simetria e baixas porcentagens de acertos em testes de transitividade e equivalência. Nos participantes que não formaram classes de equivalência, foi replicado o efeito indesejável de que a resposta "não" era emitida para todos os compostos contendo combinações envolvendo estímulos que não foram apresentados juntos durante a

fase de treino.

No segundo experimento de Tovar e Torres (2012), foram feitas seis execuções do modelo computacional desenvolvido, sendo equivalentes à simulação de seis participantes humanos. Em cada execução, os pesos iniciais das conexões foram atribuídos com valores aleatórios, conferindo heterogeneidade a cada execução. A arquitetura de rede utilizada está representada na Figura 2.

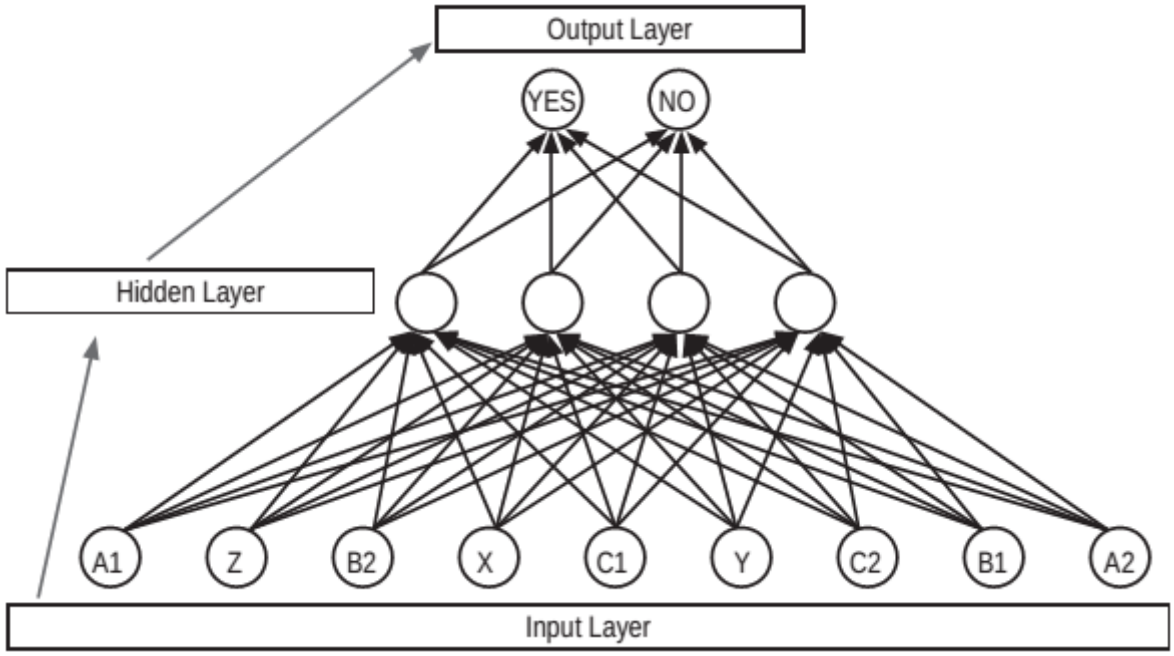

Figura 2. Arquitetura de rede utilizada por Tovar e Torres (2012), conforme apresentado no trabalho original.

A arquitetura de rede utilizada por Tovar e Torres (2012) continha três camadas: uma camada para entrada de dados, uma camada intermediária e uma camada de saída de dados. A camada de entrada de dados, representada na parte inferior da figura, era composta de nove unidades de ativação que podiam receber os valores 0 ou 1. Cada uma das nove unidades correspondia a um elemento: A1, B1, C1, A2, B2, C2, X, Y, Z. O valor 1 para uma unidade representava a presença de seu elemento correspondente na tentativa, enquanto o valor 0 representava a ausência de seu elemento correspondente. Assim, por exemplo, a representação computacional da apresentação do composto A1B1 era feita pela ativação da unidade referente ao A1 com valor 1, a unidade referente ao B1 com valor 1 e todas as demais unidades com valor 0, gerando o vetor de entrada correspondente à apresentação desse

composto. A camada intermediária era formada por quatro unidades e estava conectada com as demais segundo a arquitetura *feed-forward*. A camada de saída de dados continha duas unidades, relacionadas às duas respostas possíveis no procedimento *yes-no* com estímulos compostos. A representação computacional da resposta "sim" era dada pelo vetor de saída cujo valor referente à unidade "sim" era próximo de 1 e o valor referente à unidade "não" era próximo de 0. A representação computacional da resposta "não" era dada pelo vetor de saída cujo valor referente à unidade "sim" era próximo de 0 e o valor referente à unidade "não" era próximo de 1.

Conforme já discutido, em simulações computacionais de desempenhos humanos é importante simular o conhecimento pré-experimental. Na simulação do comportamento de formação de classes de equivalência, isso é feito por meio do treino de classes adicionais utilizando-se um conjunto extra de estímulos (e.g., Barnes & Hampson, 1993; Lyddy & Barnes-Holmes, 2007). Para implementar o treino de classes adicionais, Tovar e Torres (2012) utilizaram três estímulos adicionais: X, Y e Z. O algoritmo *backward propagation* foi usado para treinar as relações XY, YZ e XZ. Ao final do treino, os vetores de entrada correspondentes à apresentação dos compostos XY, YZ e XZ geravam o vetor de saída correspondente à representação da resposta "sim". A relação de transitividade foi diretamente treinada para assegurar o estabelecimento de uma classe de equivalência completa anteriormente ao treino usando-se o conjunto de estímulos ABC.

Após o treino de classe adicional, foram treinadas as mesmas relações AB e BC do estudo com humanos. Assim, por exemplo, para o treino da relação AB, a partir dos vetores de entrada correspondentes à apresentação do composto A1B1 ou A2B2, era esperado que fosse gerado o vetor de saída correspondente à representação da resposta "sim". Já a partir dos vetores de entrada correspondentes à apresentação do composto A1B2 ou A2B1, era esperado que fosse gerado o vetor de saída correspondente à representação da resposta "não". O algoritmo *backward propagation* foi utilizado para modificar os pesos das conexões até que os vetores de saída gerados estivessem de acordo com o *dataset* estipulado. Na fase de teste, foi avaliada a emergência da relação AC. Seria indicativo da emergência dessa relação se a partir dos vetores de entrada correspondentes à apresentação do composto A1C1 ou A2C2 fosse gerado o vetor de saída correspondente à representação da resposta "sim". Já a partir dos vetores de entrada correspondentes à apresentação do composto A1C2 ou A2C1, era esperado que fosse gerado o vetor de saída correspondente à representação da resposta "não". As relações de simetria e equivalência não foram testadas porque o modelo não fazia representação da localização espacial. Sendo assim, não havia distinção entre uma relação e

sua relação simétrica, dado que ambas possuíam os mesmos vetores de entrada. Como resultados, obteve-se que cinco das seis execuções demonstraram emergência da relação de transitividade AC e, portanto, apresentaram formação de classes de equivalência.

Para avaliar a importância da simulação do conhecimento pré-experimental (treino de classe adicional) para o modelo desenvolvido, os autores repetiram o procedimento de simulação excluindo-se as etapas referentes ao treino de classe adicional (i.e., excluindo-se o pré-treino das relações XY, YZ e XZ). Apenas as relações AB e BC foram treinadas, testando-se em seguida a emergência da relação AC. Como resultados, obteve-se que a remoção do treino de classe adicional fez com que a resposta "sim" não necessariamente fosse relacionada aos estímulos contendo elementos de mesma classe, bem como a resposta "não" aos estímulos contendo elementos de classes diferentes.

O modelo proposto por Tovar e Torres (2012) não diferencia as funções de estímulo modelo e de estímulo de comparação e, portanto, não foi preciso utilizar um recurso computacional semelhante ao duplicador do marcador do modelo próprio das redes derivadas do RELNET. Os vetores de saída eram gerados exclusivamente pela informação dos estímulos presentes a cada tentativa. Portanto, os desempenhos obtidos na fase de teste podem ser considerados como respostas emergentes e o modelo proposto é uma alternativa ao RELNET. Contudo, o procedimento *yes-no* com estímulos compostos, em humanos, pode gerar efeitos indesejáveis. Conforme discutido por Fields et al. (1997), o fato dos participantes terem que escolher entre as palavras "sim" e "não" nos testes de transitividade e equivalência fez com que alguns participantes escolhessem "não" em todas as tentativas dos testes de transitividade, uma vez que as relações testadas eram novas. Idealmente, seria interessante que simulações computacionais fizessem uso de um procedimento que, em humanos, não apresentasse o tipo de problema apontado por Fields et al. (1997). Por esse motivo, seria vantajoso que fosse avaliada a possibilidade de se utilizar, em simulações, um procedimento que não diferencia as funções de estímulo modelo e de estímulo de comparação, mas que não apresente esse tipo de problema.

Tovar e Torres (2012) sugerem que, além do procedimento *yes-no*, o procedimento *go/no-go* com estímulos compostos (e.g., Debert, Matos & McIlvane, 2007; Perez, Campos & Debert, 2009; Debert, Huziwara, Faggiani, De Mathis & Mcilvane, 2009; Campos, Debert, Barros & McIlvane, 2011; Grisante et al., 2013) seria uma alternativa para o procedimento MTS no estudo de formação de classes de equivalência, à medida que o procedimento *go/no-go* com estímulos compostos também não diferencia as funções de estímulo modelo e de estímulo de comparação.

A principal diferença entre os procedimentos *yes-no* (e.g. Fields et al., 1997) e *go/no-go* com estímulos compostos é a resposta exigida do participante. No primeiro, diante dos estímulos da mesma classe, o participante deve emitir a resposta "sim" e não emitir a resposta "não", enquanto diante dos estímulos de classes distintas o participante deve emitir a resposta "não" e não emitir a resposta "sim". Além disso, no procedimento *yes-no* existe um rótulo específico para cada uma das respostas. Já no procedimento *go/no-go* com estímulos compostos, diante de compostos contendo estímulos da mesma classe, o participante deve emitir a topografia de resposta definida previamente pelo experimentador como sendo a resposta *go* (e.g., clicar com o mouse, apertar uma tecla etc.), enquanto diante de compostos contendo estímulos de classes distintas o participante deve emitir a resposta definida como *no-go* pelo experimentador, isto é, deve emitir qualquer outra resposta exceto a resposta definida como *go*.

No estudo de Debert, Matos e McIlvane (2007) foi avaliada a possibilidade de se estabelecer classes de equivalência utilizando-se o procedimento *go/no-go* com estímulos compostos em humanos. Participaram do experimento seis adultos. Foram treinadas as relações AB e BC para três conjuntos de estímulos. A cada tentativa, na tela de um computador era exibido um composto contendo dois elementos. Diante de compostos contendo elementos de mesma classe, a resposta de clicar na tela do computador (definida como a resposta *go*) era seguida de reforço, enquanto a mesma resposta diante de compostos contendo elementos de classes diferentes não era seguida de consequências programadas. Após atingido o critério de mudança de fase, foi testada, em extição, a emergência de relações de simetria (BA e CB), transitividade (AC) e equivalência (CA). Como resultados, obteve-se que quatro dos seis participantes apresentaram todas as relações emergentes testadas, demonstrando que o procedimento *go/no-go* com estímulos compostos é uma alternativa ao MTS para a formação de classes de equivalência.

Diferentemente do procedimento *yes-no* com estímulos compostos, a literatura aponta que o procedimento *go/no-go*, em humanos, não apresenta problemas semelhantes aos encontrados no procedimento *yes-no* com relação ao fato dos participantes responderem "não" para todos os compostos em testes de transitividade (e.g., Fields et al., 1997), não sendo necessários pré-treinos adicionais (e.g., Fields et al., 2009) para assegurar a formação de classes de equivalência (e.g., Debert, Huziwara, Faggiani, De Mathis & Mcilvane, 2009; Debert, Matos & McIlvane, 2007; Grisante et al., 2013; Perez, Campos & Debert, 2009). Considerando que idealmente é desejável trabalhar em simulações computacionais com procedimentos que se mostraram bem sucedidos com humanos, seria vantajoso avaliar a

possibilidade de simular a formação de classes de equivalência com o procedimento *go/no-go* com estímulos compostos.

Outro aspecto que torna vantajosa a utilização do procedimento *go/no-go* com estímulos compostos em simulações computacionais diz respeito à representação computacional. No procedimento *yes-no* com estímulos compostos é preciso fazer a representação de duas respostas distintas: a resposta "sim" e a resposta "não". As repostas são independentes e, portanto, cada uma precisa ter uma representação computacional independente. Por esse motivo, Tovar e Torres (2012) desenvolveram uma arquitetura de rede na qual a camada de saída de dados possui duas unidades de ativação. A camada de saída de dados (na parte superior da Figura 2) possui uma unidade de ativação referente à resposta "sim" (denotada *YES* na Figura 2) e outra unidade de ativação referente à resposta "não" (detonada *NO* na Figura 2). A representação computacional da resposta "sim" é feita por meio do vetor de saída {1, 0}. Isto é, a ativação da unidade "sim" com o valor 1 e a ativação da unidade "não" com o valor 0. A representação computacional da resposta "não" é feita de forma inversa, por meio do vetor de saída {0, 1}. Isto é, a ativação da unidade "sim" com o valor 0 e a ativação da unidade "não" com o valor 1. Apesar dessa arquitetura estar conceitualmente correta, devido à independência das respostas "sim" e "não", computacionalmente existe uma redundância: sempre que a unidade de ativação "sim" for ativada com valor 1, é esperado que a unidade de ativação "não" seja necessariamente ativada com valor 0 e, de forma inversa, sempre que a unidade de ativação "sim" for ativada com valor 0, espera-se que a unidade "não" seja ativada com valor 1. Portanto, é computacionalmente redundante que existam duas unidades de ativação na camada de saída. Seria mais parcimonioso e simplificaria o procedimento computacional de treino se a camada de saída de dados tivesse apenas uma unidade de ativação. Isso é possível ao se desenvolver uma arquitetura de rede utilizando-se o procedimento *go/no-go* com estímulos compostos. Nesse procedimento, a resposta *no-go* é definida pela pela emissão de qualquer resposta diferente da resposta definida como *go*. Por existir uma dependência entre as respostas *go* e *no-go*, seria conceitualmente correto propor uma arquitetura contendo apenas uma unidade de ativação na camada de saída de dados. A ativação da única unidade da camada de saída com valor 1 seria a representação computacional da resposta *go* e a ativação dessa unidade com valor 0 seria a representação computacional da emissão de qualquer resposta diferente de *go*, em outras palavras, da resposta *no-go*. Essa arquitetura de rede contendo apenas uma unidade de ativação na camada de saída de dados evitaria redundâncias computacionais, facilitaria o processo computacional de treino e estaria conceitualmente correta com as definições das

respostas *go* e *no-go*.

O presente trabalho se propôs a avaliar a possibilidade de utilizar RNAs para simular o comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos. Para isso, o método de Tovar e Torres (2012) foi tomado como base para se desenvolver um modelo computacional adaptado ao procedimento *go/no-go* com estímulos compostos.

EXPERIMENTO 1

No presente estudo, a programação das RNAs foi feita utilizando-se a Linguagem C (Ritchie, Kernighan & Lesk, 1975) e a *Fast Artificial Neural Network Library* (FANN) (Nissen, 2003), cuja última atualização foi no ano de 2015. A Linguagem C em conjunto com a FANN provê uma plataforma de programação com grande compatibilidade para os sistemas operacionais mais atuais e ampla portabilidade, necessitando apenas de um compilador da Linguagem C e, por isso, funciona em *Windows*, *Linux*, *Unix*, *Android*, entre outros. O Experimento 1 teve como objetivo assegurar que o uso dessas ferramentas para a programação das RNAs não implicaria em mudanças significativas nos resultados, que impossibilitassem comparaçãoes com estudos que utilizaram outras ferramentas. Para isso, foi feita uma replicação sistemática do Experimento 2 de Tovar e Torres (2012), mantendo-se todos os parâmetros originais do estudo e alterando-se apenas a utilização da Linguagem C e FANN para a programação das RNAs. Em Tovar e Torres (2012), não foi descrito qual recurso computacional foi utilizado para a programação das RNAs.

Método

Todas as etapas de simulação computacional do estudo foram feitas utilizando-se um notebook Dell Inspiron 14R 5421, que continha processador Intel Core i7-3537U, NVIDIA GeForce GT 730M DDR3 de 2GB dedicados e 8 GB RAM.

Foi utilizada a mesma arquitetura de rede de Tovar e Torres (2012), representada na Figura 2. As etapas de treino consistiram no pré-treino das relações XY, YZ e XZ para uma classe de estímulos (i.e., XYZ), seguida do treino das relações AB e BC para duas classes de estímulos (i.e., A1B1C1 e A2B2C2). Na fase de teste, foi avaliada a emergência da relação AC. Conforme no trabalho original, durante o treino, os pesos iniciais foram atribuídos com valores aleatórios entre 0 e 1 e o algoritmo *backward propagation* foi utilizado com *learning rate* 0,3. Durante o teste, esse algoritmo foi desativado e os pesos das conexões foram mantidos constantes. O código fonte do modelo computacional desenvolvido para a replicação sistemática de Tovar e Torres (2012) está disponível no Apêndice A.

Resultados

O critério de mudança de fase durante o treino foi atingido, em média, em 431 *epochs*. Cada *epoch* correspondeu ao conjunto de iterações em que cada relação de treino foi apresentada uma vez. Considerando que foram treinadas 11 relações, o critério de mudança de fase, portanto, foi atingido em média após 4745 iterações.

Na Figura 3, encontram-se os resultados da replicação sistemática do experimento de Tovar e Torres (2012) utilizando-se Linguagem C e FANN para a programação das RNAs. Cada gráfico corresponde a uma execução do modelo computacional. No eixo horizontal estão os compostos apresentados. No eixo vertical estão os valores de ativação da unidade correspondente à Resposta *YES* e da unidade correspondente à Resposta *NO*, na camada de saída de dados. A linha tracejada é o critério para a representação da Resposta *YES* ou da Resposta *NO*. Conforme em Tovar e Torres (2012), valores iguais ou superiores a 0,85 gerados pela unidade de ativação *YES* foram considerados como representação da Resposta *YES*, enquanto valores iguais ou superiores a 0,85 gerados pela unidade de ativação *NO* foram considerados como representação da Resposta *NO*.

Cinco das seis execuções atingiram os critérios indicativos de respostas emergentes para a relação AC, consequentemente confirmando a formação de classes de equivalência. A Execução 2 apresentou valores próximos do critério, apesar de não o ter atingido.

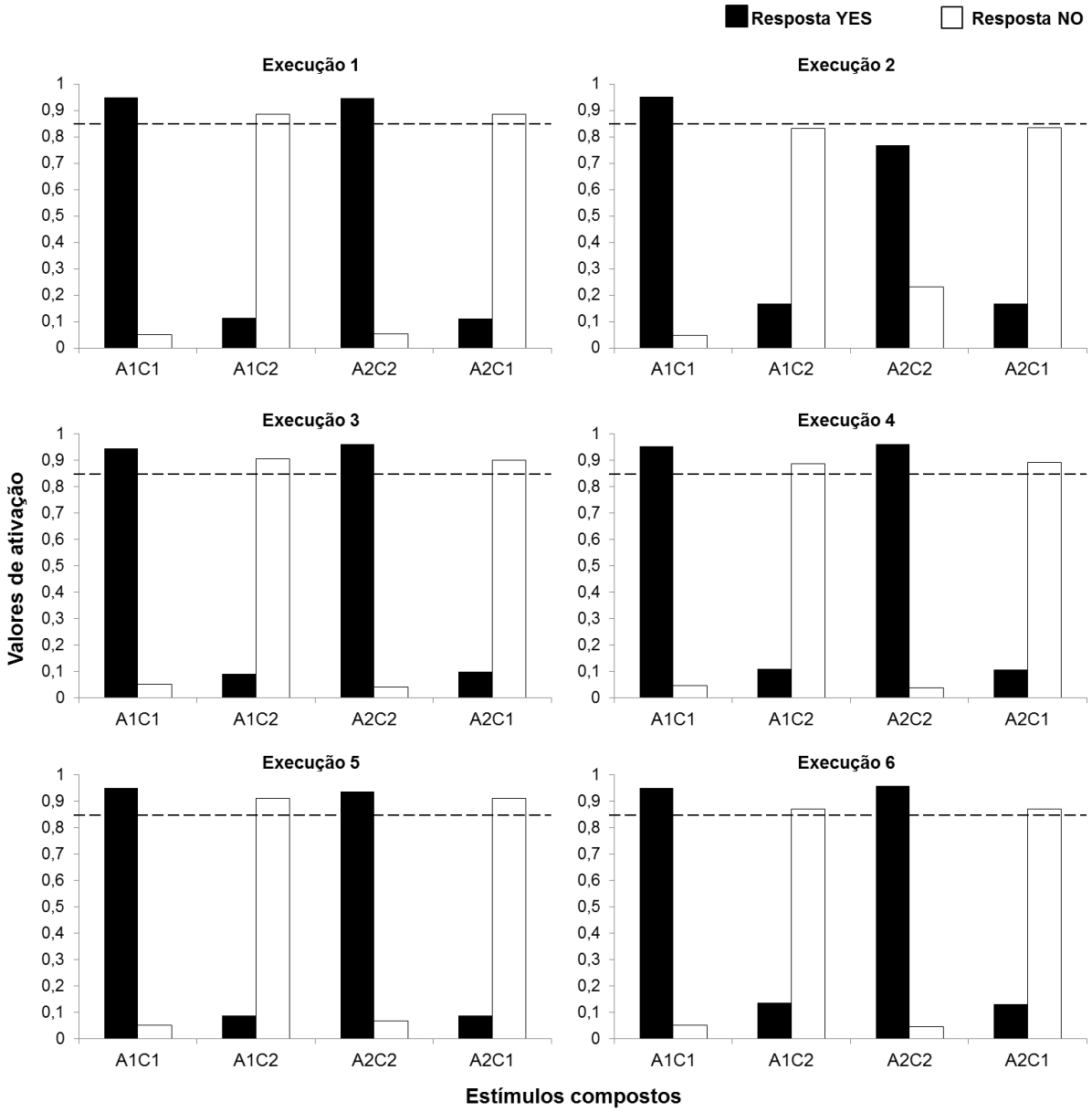

Figura 3. Valores de ativação das Respostas "YES" e "NO" para cada estímulo composto apresentado em cada uma das execuções na fase de teste do Experimento 1.

Discussão

Os resultados obtidos foram muito semelhantes aos resultados de Tovar e Torres (2012), que também encontraram que cinco das seis execuções atingiram os critérios indicativos de respostas emergentes para a relação AC. Isso demonstra que a programação utilizando-se Linguagem C e FANN não alterou o funcionamento das RNAs. Esses resultados

estão de acordo com a premissa de que os únicos aspectos que efetivamente determinam o funcionamento de uma RNA são a arquitetura de rede e os algoritmos matemáticos utilizados. Portanto, pode-se afirmar que a mudança para a Linguagem C e FANN não implica em alterações nos resultados finais, possibilitando que o presente estudo seja comparado com trabalhos que utilizaram outras ferramentas para a programação das RNAs.

Para atingir o critério de mudança de fase, o modelo programado usando-se Linguagem C e FANN necessitou de 4745 iterações em média, número significativamente menor do que as 13898 iterações necessárias em média para atingir o critério no trabalho de Tovar e Torres (2012). Isso significa que os resultados foram gerados mais rapidamente quando foi utilizada a Linguagem C e a FANN. Contudo, ressalta-se que os resultados finais da fase de teste foram os mesmos. A mudança de ferramenta não implicou em mudança nos resultados finais, fazendo-se apenas com que estes resultados fossem atingidos mais rapidamente. A maior rapidez com que a Linguagem C e FANN atingem os resultados sugere que essas ferramentas fazem melhor uso dos recursos computacionais disponíveis em comparação com outras ferramentas.

EXPERIMENTO 2

O Experimento 2 teve como objetivo avaliar a viabilidade de se utilizar RNAs para simular o comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos. Para a programação das RNAs, foi utilizada a linguagem que se mostrou efetiva no Experimento 1. O modelo computacional foi desenvolvido baseando-se na arquitetura e nos algoritmos utilizados por Tovar e Torres (2012). As etapas de treino e teste foram repetidas em seis execuções, o que correspondeu à simulação de seis participantes humanos. O procedimento consistiu em uma fase de treino de classe adicional para simular o conhecimento pré-experimental, conforme feito por Tovar e Torres (2012), uma fase de treino das relações AB e BC, e uma fase de teste de emergência da relação AC.

Método

Inicialmente foi feito o treino de uma classe adicional contendo três estímulos (XYZ), conforme realizado por Tovar e Torres (2012). Foram estabelecidas as relações XY, YZ e XZ. A relação de transitividade XZ foi diretamente treinada para assegurar o estabelecimento da classe de equivalência XYZ. Após concluído o treino de classe adicional, foram treinadas as relações AB e BC para dois conjuntos de estímulos (i.e., A1B1C1 e A2B2C2). Por fim, foi testada a emergência da relação AC. As etapas de treino e teste foram repetidas em seis execuções.

As RNAs foram programadas com arquitetura *feed-forward* contendo três camadas, baseando-se na arquitetura utilizado por Tovar e Torres (2012), conforme esquematizado na Figura 4. As unidades da camada de entrada de dados foram dispostas de forma a evitar treino de relações espaciais e sequenciais (Tovar & Torres, 2012). Isto é, a camada de entrada de dados foi desenvolvida de maneira a evitar que os vetores de saída fossem determinados pela forma com que as unidades de ativação foram organizadas (e.g. se todas as unidades da Classe 1 fossem dispostas na parte superior e da Classe 2 na parte inferior, seria possível que os vetores de saída fossem determinados por essa organização). A representação dos compostos presentes a cada tentativa foi feita na camada de entrada usando-se os valores 0 e 1. Cada unidade dessa camada correspondia a um estímulo experimental em específico. A ativação de

uma unidade com valor 1 representava a presença de seu estímulo correspondente na tentativa, enquanto a ativação de uma unidade com valor 0 representava a ausência de seu estímulo correspondente. Assim, por exemplo, para representar a presença do composto A1B1 em uma tentativa, a unidade correspondente ao estímulo A1 e a unidade correspondente ao estímulo B1 recebiam o valor 1 e todas as outras unidades recebiam o valor 0, gerando um vetor de entrada. A representação da resposta *go* ou da resposta *no-go* a cada tentativa foi feita na camada de saída de dados, cuja unidade de ativação poderia gerar valores entre 0 e 1. Adaptando-se o critério utilizado em Tovar e Torres (2012), foi considerado como representação da resposta *go* a ativação da unidade da camada de saída de dados com valor igual ou superior a 0,85. De forma inversa, foi considerado como representação da resposta *no-go* a ativação da unidade da camada de saída de dados com valor igual ou inferior a 0,15.

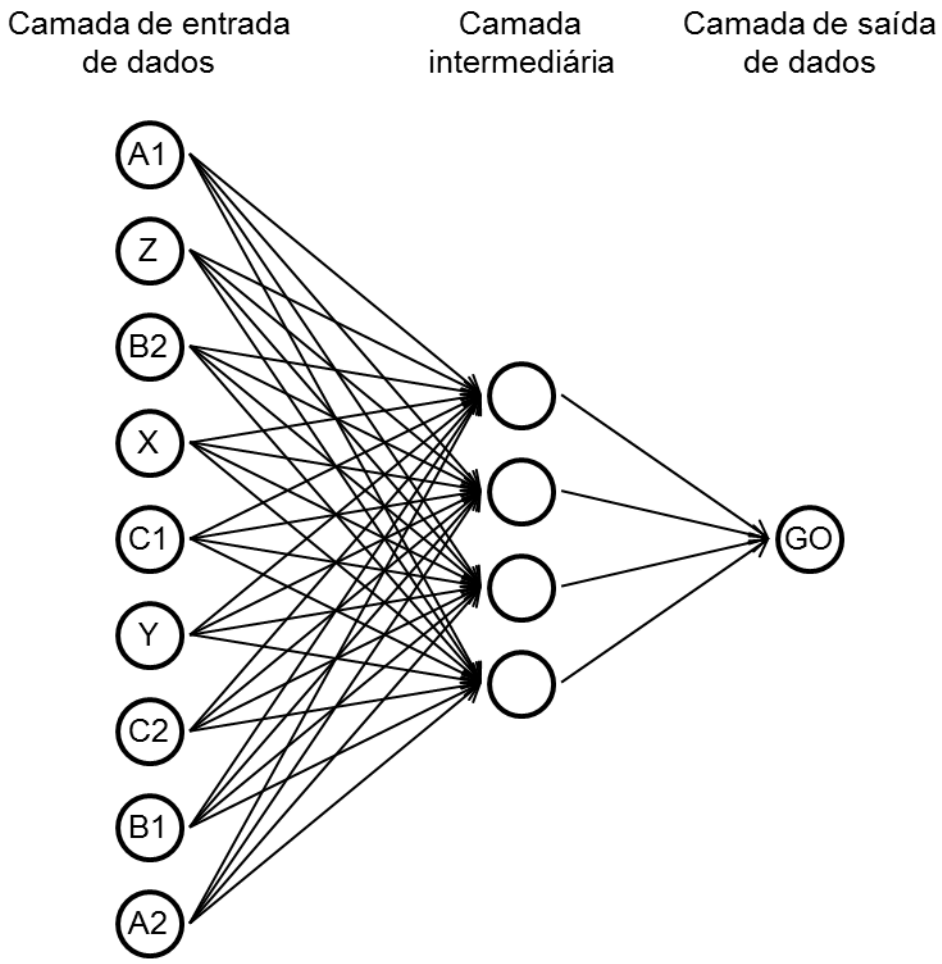

Figura 4. Arquitetura de rede utilizada para as etapas de treino e teste que envolveram treino de classe adicional.

Os vetores de entrada e seus vetores de saída correspondentes considerados corretos durante a fase de treino, ou esperados durante a fase de teste, estão na Tabela 1. Assim, por exemplo, a partir do vetor de entrada correspondente à representação da presença do composto A1B1, era considerado correto que fosse gerado o vetor de saída correspondente à resposta *go*. Já a partir do vetor de entrada correspondente à representação da presença do composto A1B2, era considerado correto que fosse gerado o vetor de saída correspondente à resposta *no-go.*

Tabela 1 – Vetores de entrada e seus vetores de saída correspondentes considerados corretos na fase de Treino, ou esperados na fase de Teste, havendo treino de classe adicional.

Fase	**Composto**	**Vetores de entrada**									**Vetores de saída**
		A1	**Z**	**B2**	**X**	**C1**	**Y**	**C2**	**B1**	**A2**	**Go/No-go**
Treino	XY	0	0	0	1	0	1	0	0	0	1
Treino	YZ	0	1	0	0	0	1	0	0	0	1
Treino	XZ	0	1	0	1	0	0	0	0	0	1
Treino	A1B1	1	0	0	0	0	0	0	1	0	1
Treino	A1B2	1	0	1	0	0	0	0	0	0	0
Treino	B1C1	0	0	0	0	1	0	0	1	0	1
Treino	B1C2	0	0	0	0	0	0	1	1	0	0
Treino	A2B2	0	0	1	0	0	0	0	0	1	1
Treino	A2B1	0	0	0	0	0	0	0	1	1	0
Treino	B2C2	0	0	1	0	0	0	1	0	0	1
Treino	B2C1	0	0	1	0	1	0	0	0	0	0
Teste	A1C1	1	0	0	0	1	0	0	0	0	1
Teste	A1C2	1	0	0	0	0	0	1	0	0	0
Teste	A2C2	0	0	0	0	0	0	1	0	1	1
Teste	A2C1	0	0	0	0	1	0	0	0	1	0

Os demais parâmetros de configuração das RNAs, descritos a seguir, foram adaptados de Tovar e Torres (2012). Os pesos inicias das conexões entre as unidades de ativação foram gerados aleatoriamente por meio de uma distribuição uniforme de intervalo entre 0 e 1, fazendo com que as execuções fossem diferentes umas das outras. Foi utilizada a função sigmoidal como função de ativação (ver Haykin, 2001). Na fase de treino, quando o vetor de saída apresentado não correspondia perfeitamente ao vetor de saída considerado correto na

tentativa, foi usado o algoritmo de correção *backward propagation*, contendo os parâmetros de *learning rate* 0,3 e *learning momentum* zero. O algoritmo alterava os valores dos pesos das conexões, fazendo com que os vetores de saída gerados fossem cada vez mais próximos do que era considerado correto. A fase de treino foi finalizada quando o critério EQM $\leq$ 0,0025 foi atingido.

Na fase de teste, foi avaliada a emergência da relação de transitividade AC. O algoritmo de correção foi desativado e os pesos das conexões foram mantidos constantes. Era esperado que fosse gerado o vetor de saída correspondente à representação computacional da resposta *go* a partir dos vetores de entrada correspondentes aos compostos A1C1 ou A2C2. Também era esperado que fosse gerado o vetor de saída correspondente à representação computacional da resposta *no-go* a partir dos vetores de entrada correspondentes aos compostos A1C2 ou A2C1. Dado que a arquitetura da RNA foi inspirada em um modelo que não considera a localização espacial, conforme em Tovar e Torres (2012), os vetores de entrada de uma relação e de sua relação simétrica eram os mesmos. Por isso, não foram feitos testes de relações de simetria. O código fonte do modelo computacional utilizado para simular o comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos está disponível no Apêndice B.

Resultados

O critério de mudança de fase durante o treino foi atingido, em média, em 526 *epochs*. Considerando que foram treinadas 11 relações, o critério de mudança de fase, portanto, foi atingido em média após 5781 iterações.

Na Figura 5 estão os resultados das simulações computacionais do comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos para as seis execuções. Os resultados são os desempenhos na fase de teste de emergência da relação AC, após o treino de classe adicional e dos treinos das relações AB e BC. A resposta *go* corresponde a valores iguais ou superiores a 0,85 (linha tracejada superior), enquanto a resposta *no*-go corresponde a valores iguais ou inferiores a 0,15 (linha tracejada inferior).

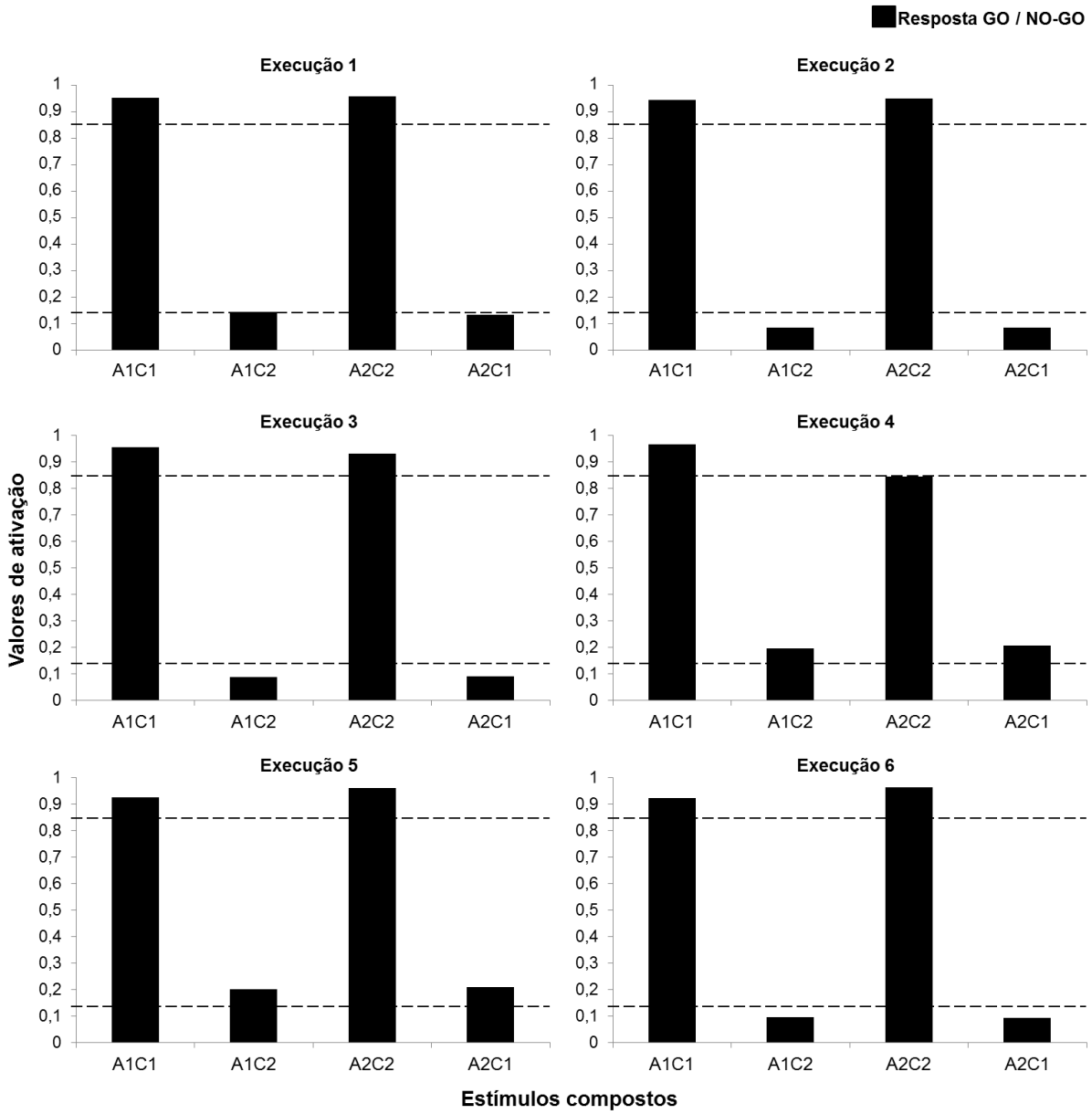

Figura 5. Valores de ativação para cada composto da fase de teste em cada execução do Experimento 2. Valores acima da linha tracejada superior correspondem à representação da resposta *go*, enquanto valores abaixo da linha tracejada inferior correspondem à representação da resposta *no-go*.

Quatro das seis execuções atingiram os critérios indicativos da emergência da relação AC e da formação das classes de equivalência A1B1C1 e A2B2C2. A Execução 4 não atingiu o critério da resposta *go* para o vetor de entrada correspondente à representação computacional da apresentação do composto A2C2, nem da resposta *no-go* para as apresentações dos compostos A1C2 e A2C1. A Execução 5 atingiu os critérios da resposta *go* para as apresentações dos compostos A1C1 e A2C2, mas não atingiu os critérios da resposta

no-go para os compostos A1C2 e A2C1.

Discussão

O Experimento 2 avaliou a possibilidade de se utilizar RNAs para simular o comportamento de formação de classes de equivalência usando-se o procedimento *go/no-go* com estímulos compostos. Foi encontrado que quatro execuções atingiram os critérios para a aceitação dos resultados como respostas emergentes para a relação AC, confirmando a formação das classes de equivalência A1B1C1 e A2B2C2. O fato de nem todas as execuções atingirem os critérios para a formação de classes de equivalência assemelha-se aos resultados encontrados nos experimentos com humanos, nos quais nem sempre todos os participantes demonstram formação de classes de equivalência (e.g., Debert, Matos & McIlvane, 2007; Perez, Campos & Debert, 2009; Debert, Huziwara, Faggiani, De Mathis & Mcilvane, 2009; Campos, Debert, Barros & McIlvane, 2011; Grisante et al., 2013).

O modelo computacional avaliado no presente estudo não diferencia as funções de estímulo modelo e de estímulo de comparação, pois simula as tentativas de treino e teste de relações condicionais usando o procedimento *go/no-go* com estímulos compostos. Não foi preciso utilizar um recurso computacional semelhante ao duplicador do marcador do modelo das redes derivadas do RELNET. Os desempenhos na fase de teste foram determinados unicamente pela informação sobre os estímulos presentes a cada tentativa e, assim, pode-se considerar esses desempenhos como respostas emergentes. Portanto, o modelo proposto é uma alternativa ao RELNET para o estudo de relações de equivalência usando-se simulações computacionais.

Os resultados obtidos foram semelhantes aos resultados encontrados em Tovar e Torres (2012), no qual foi utilizado um modelo computacional que simulava o comportamento de formação de classes de equivalência por meio do procedimento *yes-no* com estímulos compostos, sem diferenciar as funções de estímulo modelo e estímulo de comparação. Das seis execuções que passaram pelo treino de classe adicional, cinco atingiram os critérios para aceitação dos resultados como respostas emergentes para a relação AC e, consequentemente, demonstraram formação de classes de equivalência. Assim, os resultados obtidos no presente trabalho somam-se aos resultados de Tovar e Torres (2012) enquanto evidências de que é possível desenvolver modelos computacionais para simulação do

comportamento de formação de classes de equivalência sem diferenciar as funções de estímulo modelo e de estímulo de comparação.

Todas as execuções do Experimento 2 passaram pelo treino de classe adicional, conforme comumente é realizado em experimentos envolvendo simulações computacionais do comportamento de formação de classes de equivalência usando-se procedimentos diferentes do *go/no-go* com estímulos compostos (e.g., Barnes & Hampson, 1993; Lyddy & Barnes-Holmes, 2007; Tovar & Torres, 2012). Contudo, ainda não se sabe se esse treino de classe adicional é uma etapa importante também para simulações que utilizam o procedimento *go/no*-go com estímulos compostos. Seria importante avaliar se na ausência desse treino, os resultados das simulações utilizando o modelo computacional desenvolvido no presente trabalho permaneceriam os mesmos ou se da mesma forma que em Tovar e Torres (2012) o treino adicional se mostraria essencial para a produção de classes de equivalência.

EXPERIMENTO 3

Estudos que utilizaram procedimentos diferentes do *go/no-go* com estímulos compostos para a simulação do comportamento de formação de classes de equivalência apontaram que o treino de classes adicionais é uma etapa importante da simulação (e.g. Barnes & Hampson, 1993; Lyddy & Barnes-Holmes, 2007; Tovar & Torres, 2012). É discutido que as classes de equivalência adicionais, estabelecidas anteriormente ao experimento, constituem-se em um análogo à experiência pré-experimental que participantes humanos possuem:

> Para que a rede seja capaz de derivar relações na ausência de treino explícito (e.g. dado A1-B1 e B1-C1 produzir A1-C1 sem treino), a rede precisou ser exposta a pré-treinos explícitos dessas relações [derivadas]. Isso efetivamente simula a história pré-experimental de desenvolvimento de equivalência em um participante humano; é assumido que um humano com capacidades linguísticas possa derivar A1-C1 em laboratório porque já foi feito reforçamento explícito para esse comportamento [de derivar novas relações] com outras classes no passado. (Lyddy & Barnes-Holmes, p.17, 2007)

Baseando-se no que comumente é feito na área, todas as execuções do Experimento 2 passaram pelo treino de classe adicional e, como resultados, obteve-se que foi possível simular o comportamento de formação de classes de equivalência. Contudo, é importante avaliar se o treino de classe adicional é de fato uma etapa importante para a simulação desse comportamento quando o procedimento *go/no-go* com estímulos compostos é utilizado. Tovar e Torres (2012), após simulação bem sucedida de formação de classes de equivalência pelo procedimento *yes-no* com treino de classe adicional, realizaram um estudo que demonstrou que as classes de equivalência não são estabelecidas sem o treino de classe adicional. Na mesma direção, o Experimento 3 do presente estudo teve como objetivo avaliar a possibilidade de simular formação de classes de equivalência com o procedimento *go/no-go* com estímulos compostos sem o treino de classe adicional.

Método

Foram realizados os mesmos procedimentos de treino e teste do Experimento 2, com a exceção de que não houve treino de classe adicional. Sendo assim, foram treinadas apenas as relações AB e BC, e foi testada a emergência da relação AC em seis execuções. Foi utilizado o mesmo modelo computacional desenvolvido para o Experimento 2 (Apêndice B). A arquitetura de rede foi alterada para não conter unidades de ativação referentes aos estímulos utilizados no treino de classe adicional (X, Y e Z), representada na Figura 6.

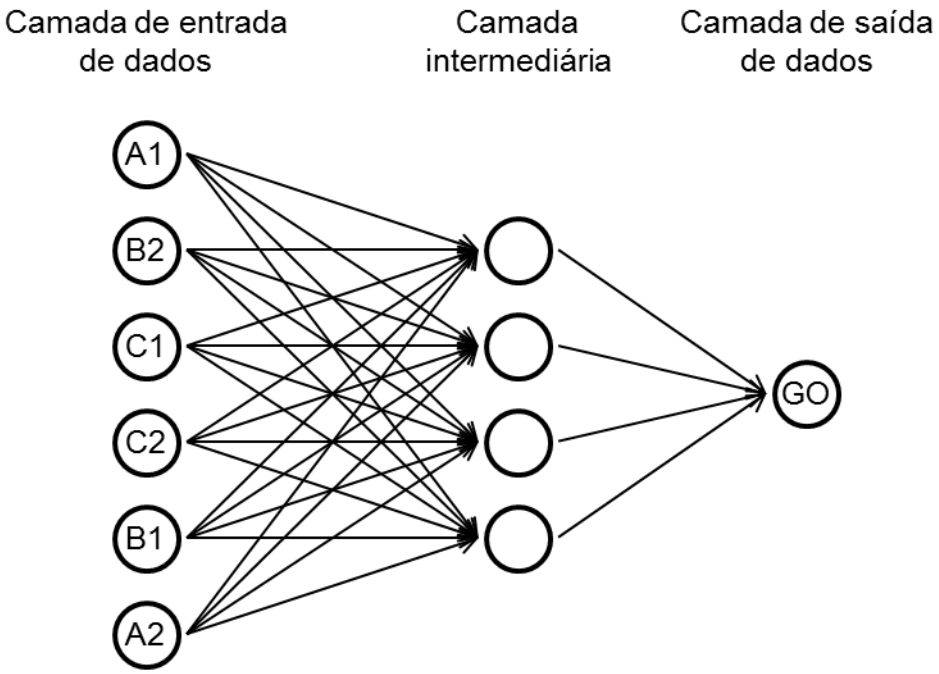

Figura 6. Arquitetura de rede utilizada para as etapas de treino e teste que não envolveram treino de classe adicional.

Os vetores de entrada e seus vetores de saída correspondentes considerados corretos durante a fase de treino, ou esperados durante a fase de teste, foram os mesmos do Experimento 1, diferindo apenas com relação a não haver representações para os estímulos X, Y e Z, nem etapas relacionadas ao treino de classe adicional. Na Tabela 2 estão os vetores de entrada e saída utilizados em cada fase.

Tabela 2 – Vetores de entrada e seus vetores de saída correspondentes considerados corretos na fase de Treino, ou esperados na fase de Teste, sem haver treino de classe adicional.

Fase	Composto	Vetores de entrada						Vetores de saída
		A1	B2	C1	C2	B1	A2	Go/No-go
Treino	A1B1	1	0	0	0	1	0	1
Treino	A1B2	1	1	0	0	0	0	0
Treino	B1C1	0	0	1	0	1	0	1
Treino	B1C2	0	0	0	1	1	0	0
Treino	A2B2	0	1	0	0	0	1	1
Treino	A2B1	0	0	0	0	1	1	0
Treino	B2C2	0	1	0	1	0	0	1
Treino	B2C1	0	1	1	0	0	0	0
Teste	A1C1	1	0	1	0	0	0	1
Teste	A1C2	1	0	0	1	0	0	0
Teste	A2C2	0	0	0	1	0	1	1
Teste	A2C1	0	0	1	0	0	1	0

Resultados

O critério de mudança de fase durante o treino foi atingido, em média, em 1092 *epochs*. Considerando que foram treinadas 8 relações, o critério de mudança de fase, portanto, foi atingido em média após 8737 iterações.

Na Figura 7 estão os resultados das simulações computacionais do comportamento de formação de classes de equivalência usando-se o procedimento *go/no-go* com estímulos compostos quando não é realizado treino de classe adicional. A resposta *go* corresponde a valores iguais ou superiores a 0,85 (linha tracejada superior), enquanto a resposta *no*-go corresponde a valores iguais ou inferiores a 0,15 (linha tracejada inferior).

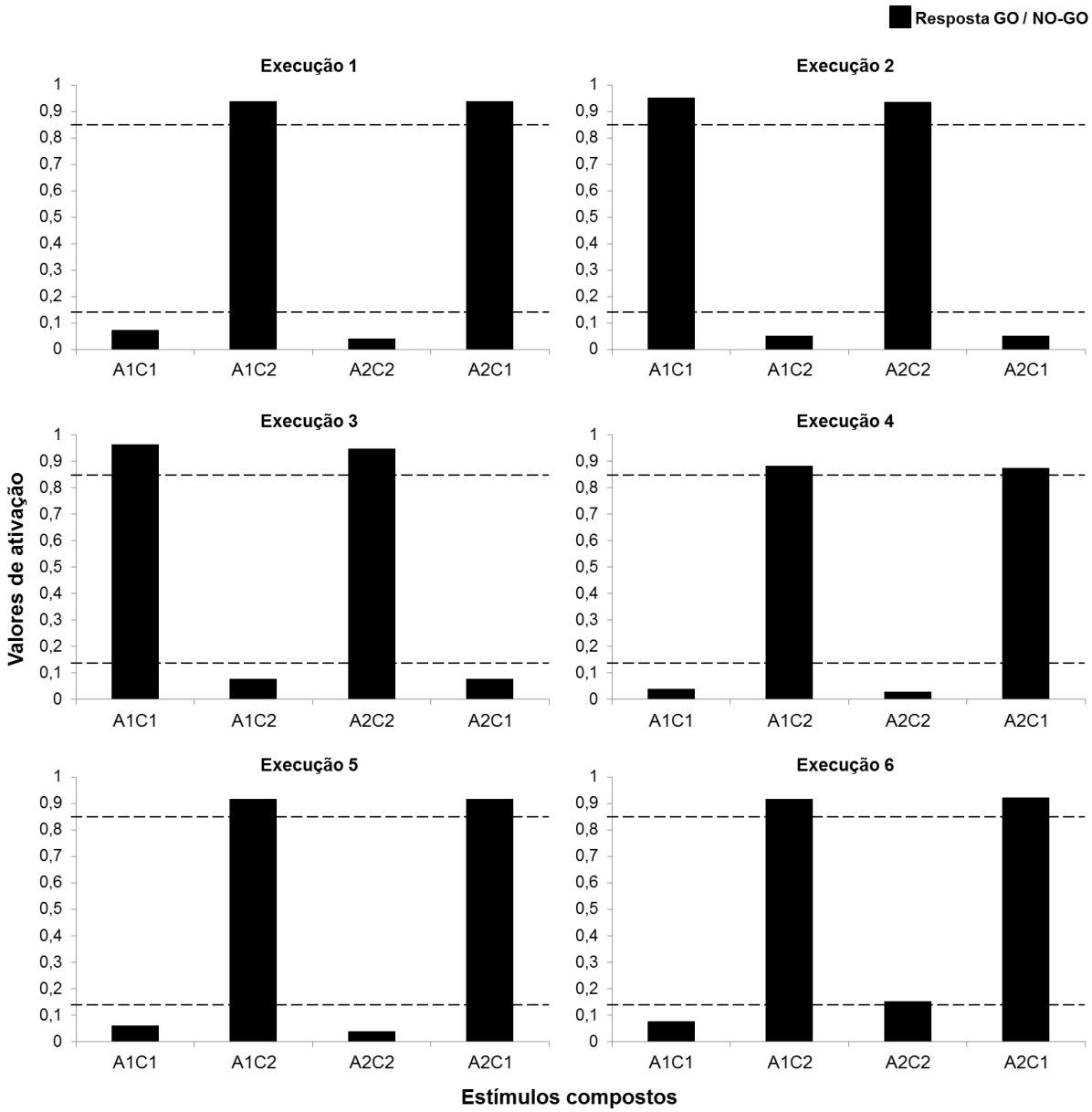

Figura 7. Valores de ativação para cada composto da fase de teste em cada execução do Experimento 3. Valores acima da linha tracejada superior correspondem à representação da resposta *go*, enquanto valores abaixo da linha tracejada inferior correspondem à representação da resposta *no-go*.

Apenas duas das seis execuções atingiram os critérios para a aceitação dos resultados como indicativos da emergência da relação AC e da formação das classes de equivalência A1B1C1 e A2B2C2. Destaca-se que as Execuções 2, 4 e 5 atingiram os critérios para a representação computacional da resposta *go* para compostos contendo elementos de classes diferentes (A1C2 e A2C1), ao passo que atingiram os critérios para a representação computacional da resposta *no-go* para compostos contendo elementos de mesma classe

(A1C1e A2C2). Esse resultado envolveria o padrão inverso ao esperado no teste de transitividade. Se, na Execução 6, tivesse sido atingido o critério da resposta *no-go* para o composto A2C2, também teria sido apresentado o padrão inverso ao esperado na fase de teste, já que isso aconteceu apenas para o composto A1C1.

Discussão

O Experimento 3 avaliou o efeito da ausência do treino de classe adicional sobre os resultados da simulação do comportamento de formação de classes de equivalência usando-se o procedimento *go/no-go* com estímulos compostos. Na fase de treino, o critério de mudança de fase foi atingido em média após 8737 iterações, número significativamente maior em comparação com as 5781 iterações em média necessárias no Experimento 2. O aumento de iterações para atingir o critério no Experimento 3 indica que o treino de classe adicional facilita o estabelecimento de relações diretamente treinadas.

Na fase de teste, foi encontrado que apenas duas das seis execuções atingiram os critérios indicativos da emergência da relação AC e da formação das classes de equivalência A1B1C1 e A2B2C2. Portanto, da mesma forma que foi indicado por Barnes e Hampson (1993), Lyddy e Barnes-Holmes (2007) e Tovar e Torres (2012) no caso dos procedimentos MTS e *yes-no*, o treino de classe adicional é essencial para simular comportamento de classe de equivalência com o procedimento *go/no-go* com estímulos compostos.

Além disso, três das execuções que não demonstraram formação de classes de equivalência apresentaram o que foi chamado de padrão inverso ao esperado na fase de teste. Isto é, os critérios para a representação computacional da resposta *go* foram atingidos para a apresentação de compostos contendo elementos de classes diferentes, ao passo que os critérios para a representação computacional da resposta *no-go* foram atingidos para a apresentação de compostos contendo elementos de mesma classe. Esses resultados são muito diferentes do que se observa em experimentos com humanos utilizando-se o procedimento *go/no-go* com estímulos compostos (e.g., Debert, Matos & McIlvane, 2007; Perez, Campos & Debert, 2009; Debert, Huziwara, Faggiani, De Mathis & Mcilvane, 2009; Campos, Debert, Barros & McIlvane, 2011; Grisante et al., 2013). Entretanto, outros estudos com simulação de formação de classes de equvalência com o procedimento *yes-no*, por exemplo, também obtiveram o padrão inverso mencionado quando o treino de classes adicionais não foi realizado (e.g.,

Tovar & Torres, 2012). Tovar e Torres (2012) levantaram a hipótese de que o padrão inverso ao esperado aparece nas execuções em que não houve treino de classe adicional porque, nos testes de transitividade, a probabilidade da resposta "sim" era a mesma da resposta "não", quando não era feito o treino de classe adicional. Isso porque, como apontado por Fields et al (2009), no caso de humanos, o treino de linha de base com os pares de estímulos A1B1 e B1C1, associados com a resposta "sim", estabece a condição para ocorrência da resposta "sim" na presença de A1C1. Entretanto, o treino de linha de base com os pares A1B2 e B2C1 também estabelece a condição para ocorrência da resposta "não" na presenca de A1C1. Assim, é possível explicar o surgimento do padrão inverso quando não há o treino adicional. Futuros estudos poderiam investigar essa possibilidade.

Sendo assim, pode-se afirmar que o treino de classe adicional é uma etapa necessária em simulações do comportamento de formação de classes de equivalência, tanto para o procedimento *go/no*-go com estímulos compostos quanto para outros procedimentos já utilizados (e.g. Barnes & Hampson, 1993; Lyddy & Barnes-Holmes, 2007; Toaver & Torres, 2012). Contudo, uma característica que ainda não foi avaliada é se o treino de classe adicional deve conter o mesmo número de elementos que as classes de equivalência que serão estabelecidas após o treino de classe adicional. Em Lyddy e Barnes-Holmes (2007), no treino de classes adicionais foram estabelecidas três classes contendo três elementos cada e, no treino de classes adicionais feito por Tovar e Torres (2012), foi estabelecida uma classe contendo três elementos. Esses dados sugerem que o treino de apenas uma classe adicional contendo três elementos é suficiente para o estabelecimento de classes de equivalência também com três elementos nas fases subsequentes do experimento. Resta saber, por exemplo, se o treino de classe adicional com três elementos permite o estabelecimento de classes de equivalência com quatro elementos ou mais.

EXPERIMENTO 4

O estabelecimento de classes contendo mais de três termos possibilita a investigação do aumento da complexidade do comportamento de formação de classes de equivalência. Sidman e Tailby (1982) discutem que o aumento da classe faz com que um número reduzido de relações de treino produzam cada vez mais relações emergentes. A partir do treino de duas relações (AB e AC), os participantes do estudo apresentaram emergência de quatro novas relações: BA, CA, BC e CB. Já a partir do treino de três relações (AB, AC e DC), aumentando-se a classe para quatro termos, os participantes demonstraram a emergência de nove relações: BA, CA, BC, CB, CD, AD, CD, DB e BD. Os autores discutem que, quanto maior a classe, maior a proporção de relações emergente por relações treinadas.

Em classes de equivalência com mais de três termos é possível também investigar o efeito de distância nodal. Fields, Adams, Verhave e Newman (1990) investigaram o efeito de distância nodal em classes com quatro termos. Foram treinadas as relações AB, BC e CD, e testadas as relações emergentes. As relações de teste foram reapresentadas repetidamente, em extinção, até que os participantes demonstrassem formação de classes de equivalência. Foi encontrado que as relações com distância de um nó (AC, CA, BD e DB) emergiram mais rapidamente do que as relações com distância de dois nós (AD e DA). Os autores discutem que a distância nodal é uma variável determinante do grau de relacionamento dos estímulos.

Estudos usando modelos computacionais para a investigação do comportamento de formação de classes de equivalência até o momento envolveram simulações de classes contendo até três elementos (e.g., Cullinan, Barnes, Hampson & Lyddy, 1994; Lyddy & Barnes- Holmes, 2007; Lyddy, Barnes-Holmes, & Hampson, 2001; Okada, Sakagami & Yamakawa, 2005; Tovar & Torres, 2012). A simulação do comportamento de formação de classes de equivalência contendo quatro ou mais elementos possibilitaria a investigação de outros aspectos do fenômeno, como efeito de distância nodal, entre outros.

Os Experimentos 2 e 3 mostraram que o treino de classe adicional com três elementos foi fundamental para a viabilidade da simulação do comportamento de formação de classes de equivalência com três elementos. Por isso, para a avaliação da possibilidade de se realizar simulações envolvendo classes com mais de três elementos, seria importante verificar se o treino de classe adicional com três elementos seria suficiente.

Em Lyddy e Barnes-Holmes (2007), no treino de classes adicionais foram estabelecidas três classes contendo três elementos cada e, no treino de classes adicionais feito por Tovar e Torres (2012), foi estabelecida uma classe contendo três elementos. Ainda não se

sabe se o treino de classe adicional deve conter o mesmo número de elementos das classes de equivalência simuladas nas etapas subsequentes.

O Experimento 4 teve como objetivo avaliar se a partir do treino de uma classe adicional contendo três elementos seria possível simular o comportamento de formação de classes de equivalência contendo quatro elementos usando o procedimento *go/no-go* com estímulos compostos. Os procedimentos de treino e teste foram semelhantes ao Experimento 2, acrescentando-se o elemento D nas classes de equivalências. Assim, no treino de classe adicional, foram estabelecidas as relações XY, YZ e XZ e, nas etapas subsequentes, foram treinadas as relações AB, BC e CD e testada a emergência das relações AC, AD e BD.

Método

Foram realizadas seis execuções do mesmo modelo computacional utilizado nos Experimentos 2 e 3 (Apêndice B), mantendo-se a programação e todos os parâmetros de treino. Foram modificados apenas a arquitetura de rede e as relações de treino e teste. O treino de classe adicional foi feito conforme no Experimento 2, treinando-se as relações XY, YZ e XZ. As etapas subsequentes simularam tentativas de treino e teste de relações condicionais envolvendo duas classes contendo quatro elementos cada (A1B1C1D1 e A2B2C2D2). Foram treinadas as relações AB, BC e CD, e foi testada a emergência das relações AC, AD e BD, conforme esquematizado na Figura 8. A arquitetura de rede utilizada foi semelhante àquela utilizada no Experimento 2, adicionando-se na camada de entrada unidades de ativação referentes aos estímulos D1 e D2. A arquitetura está representada na Figura 9. Os vetores de entrada e seus vetores de saída considerados corretos na fase de treino, ou esperados na fase de teste, estão na Tabela 3.

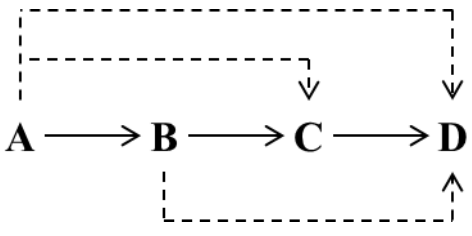

Figura 8. Relações treinadas (linhas contínuas) e testadas (linhas pontilhadas) no Experimento 4.

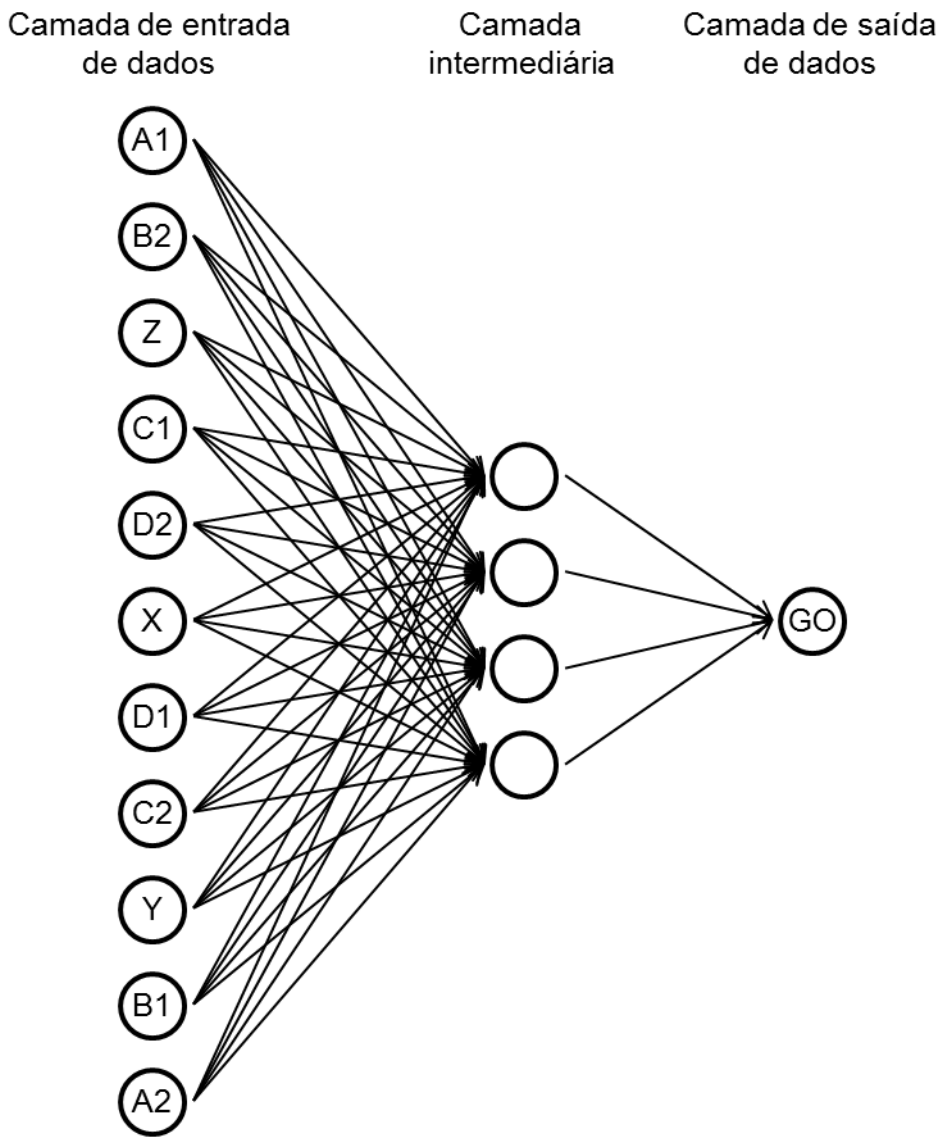

Figura 9. Arquitetura de rede das execuções que passaram pelo treino de uma classe adicional contendo três elementos.

Tabela 3 – Vetores de entrada e seus vetores de saída correspondentes considerados corretos na fase de Treino, ou esperados na fase de Teste, no Experimento 4.

Fase	Composto	Vetores de entrada											Vetores de saída
		A1	B2	Z	C1	D2	X	D1	C2	Y	B1	A2	Go/No-go
Treino	A1B1	1	0	0	0	0	0	0	0	0	1	0	1
Treino	A1B2	1	1	0	0	0	0	0	0	0	0	0	0
Treino	B1C1	0	0	0	1	0	0	0	0	0	1	0	1
Treino	B1C2	0	0	0	0	0	0	0	1	0	1	0	0
Treino	C1D1	0	0	0	1	0	0	1	0	0	0	0	1
Treino	C1D2	0	0	0	1	1	0	0	0	0	0	0	0
Treino	A2B2	0	1	0	0	0	0	0	0	0	0	1	1
Treino	A2B1	0	0	0	0	0	0	0	0	0	1	1	0
Treino	B2C2	0	1	0	0	0	0	0	1	0	0	0	1
Treino	B2C1	0	1	0	1	0	0	0	0	0	0	0	0
Treino	C2D2	0	0	0	0	1	0	0	1	0	0	0	1
Treino	C2D1	0	0	0	0	0	0	1	1	0	0	0	0
Treino	XY	0	0	0	0	0	1	0	0	1	0	0	1
Treino	YZ	0	0	1	0	0	0	0	0	1	0	0	1
Treino	XZ	0	0	1	0	0	1	0	0	0	0	0	1
Teste	A1C1	1	0	0	1	0	0	0	0	0	0	0	1
Teste	A1C2	1	0	0	0	0	0	0	1	0	0	0	0
Teste	A2C2	0	0	0	0	0	0	0	1	0	0	1	1
Teste	A2C1	0	0	0	1	0	0	0	0	0	0	1	0
Teste	A1D1	1	0	0	0	0	0	1	0	0	0	0	1
Teste	A1D2	1	0	0	0	1	0	0	0	0	0	0	0
Teste	A2D2	0	0	0	0	1	0	0	0	0	0	1	1
Teste	A2D1	0	0	0	0	0	0	1	0	0	0	1	0
Teste	B1D1	0	0	0	0	0	0	1	0	0	1	0	1
Teste	B1D2	0	0	0	0	1	0	0	0	0	1	0	0
Teste	B2D2	0	1	0	0	1	0	0	0	0	0	0	1
Teste	B2D1	0	1	0	0	0	0	1	0	0	0	0	0

Resultados

O critério de mudança de fase durante o treino foi atingido, em média, em 410 *epochs*. Considerando que foram treinadas 15 relações, o critério de mudança de fase, portanto, foi atingido em média após 6143 iterações.

Na Figura 10 estão os resultados das simulações do comportamento de formação de classes de equivalência contendo quatro elementos cada a partir do treino de uma classe adicional contendo três elementos.

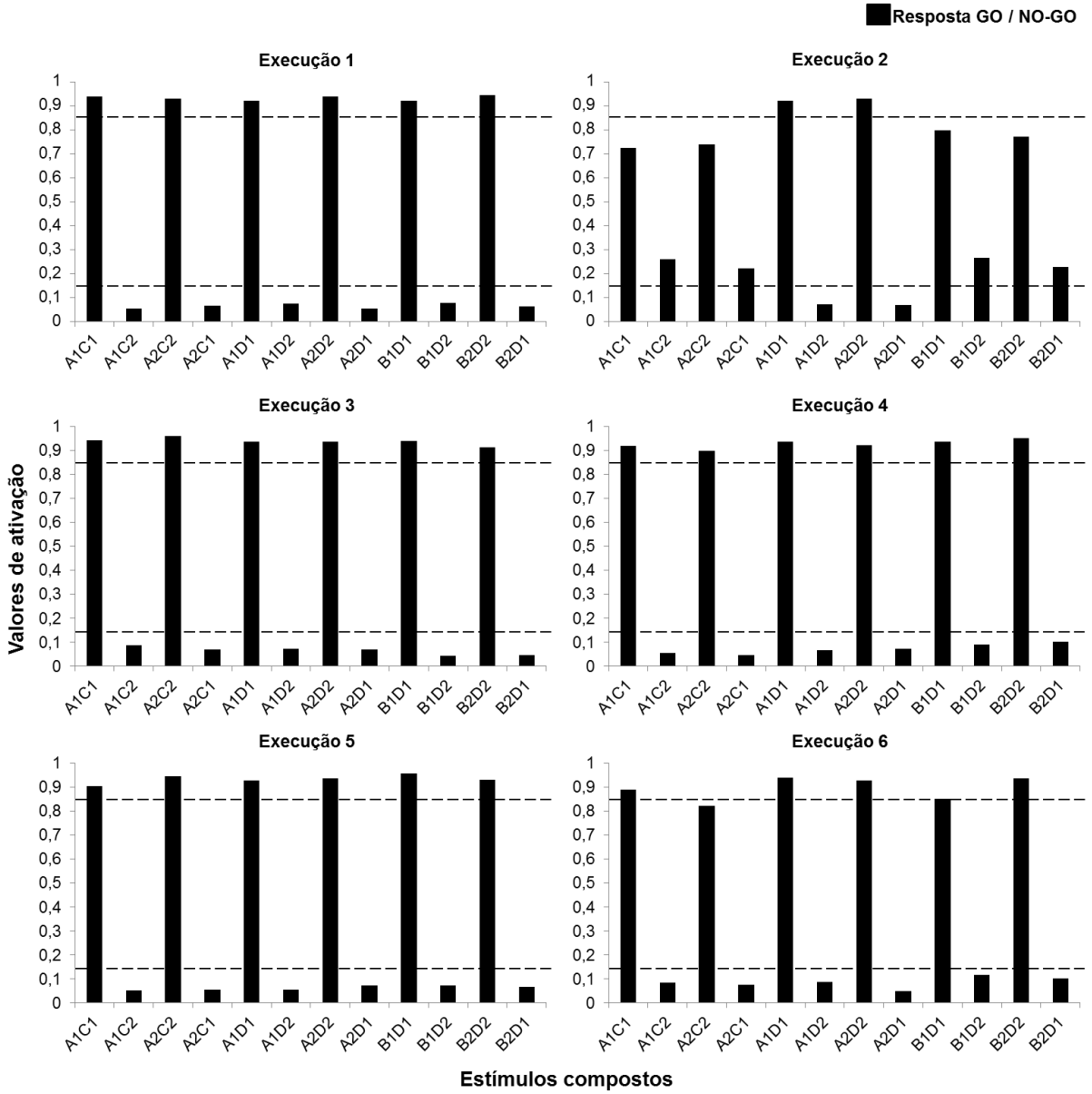

Figura 10. Valores de ativação para cada composto da fase de teste em cada execução do Experimento 4. Valores acima da linha tracejada superior correspondem à representação da resposta *go,* enquanto valores abaixo da linha tracejada inferior correspondem à representação da resposta *no-go.*

Cada gráfico corresponde a uma execução, que representa um participante humano para fins de simulação. Os resultados são os desempenhos na fase de teste de emergência das

relações AC, AD e BD, após os treinos das relações AB, BC e CD, bem como do treino de classe adicional. No eixo horizontal estão os compostos apresentados na fase de teste e no eixo vertical estão os valores gerados pela unidade de ativação correspondente à resposta *go/no-go*, na camada de saída. Valores iguais ou superiores a 0,85 (linha tracejada superior) foram considerados como a representação computacional da resposta *go*, enquanto valores iguais ou inferiores a 0,15 (linha tracejada inferior) foram considerados como a representação da resposta *no-go*.

Quatro das seis execuções atingiram os critérios para respostas emergentes das relações AC, AD e BD, indicando a formação das classes de equivalência A1B1C1D1 e A2B2C2D2 a partir do treino de uma classe adicional contendo três elementos. A Execução 2 não atingiu o critério da resposta *go* para os vetores de entrada correspondentes às representações dos compostos A1C1, A2C2, B1D1 e B2D2. Com relação à resposta *no-go*, o critério não foi atingido para as representações dos compostos A1C2, A2C1, B1D2 e B2D1. A Execução 6 não atingiu o critério da resposta *go* para as representações dos compostos A2C2 e B1D1. Em nenhuma das execuções foi encontrado o padrão inverso ao esperado, isto é, atingir os critérios para a representação computacional da resposta *go* para compostos contendo elementos de classes diferentes e atingir os critérios para a representação computacional da resposta *no-go* para compostos contendo elementos de mesma classe.

Discussão

O Experimento 4 avaliou se a partir do treino de uma classe adicional contendo três elementos seria possível simular o comportamento de formação de classes de equivalência contendo quatro elementos usando o procedimento *go/no-go* com estímulos compostos. Como resultados, obteve-se que quatro das seis execuções atingiram os critérios da emergência das relações AC, AD e BC, indicativos da formação das classes de equivalência A1B1C1D1 e A2B2C2D2. Isso mostra que o treino de uma classe adicional contendo três elementos é suficiente para a simulação do comportamento de formação de classes de equivalência contendo quatro elementos. A fase de treino do Experimento 4 foi finalizada com maior número de iterações em comparação aos demais experimentos, o que reflete o maior número de relações estabelecidas durante o treino nesse experimento.

A simulação da formação de classes contendo quatro elementos envolveu a

emergência de relações de transitividade com distância de um nó (AC e BC), bem como uma relação de transitividade com distância de dois nós (AD). Contudo, no treino de classe adicional foi estabelecido apenas uma relação de transitividade com distância de um nó (XZ), não tendo sido estabelecidas relações de transitividade com distâncias de dois ou mais nós. Sendo assim, pode-se afirmar que o treino de classes adicionais não precisa envolver o mesmo número de nós que as classes estabelecidas posteriormente às classes adicionais.

O Experimento 4 demonstrou que é possível simular a formação de classes de equivalência contendo mais de três elementos e mostra que essa simulação pode ser feita a partir do treino adicional de uma classe adicional contendo três elementos. A partir disso, futuros estudos podem investigar questões advindas do aumento da classe de equivalência. Uma dessas questões é o efeito de distância nodal (e.g., Doran & Fields, 2012; Fields, Adams, Verhave & Newman, 1990; Fields, Landon-Jimenez, Buffington & Adams, 1995). Futuros estudos que consigam criar uma representação computacional para a transferência de função poderão investigar se é possível simular o efeito de distância nodal encontrado em experimentos com humanos.

DISCUSSÃO GERAL

O presente trabalho avaliou a possibilidade de utilizar RNAs para simular o comportamento de formação de classes de equivalência usando-se o procedimento *go/no-go* com estímulos compostos.

O Experimento 1 demonstrou que a utilização da Linguagem C e FANN para programar as RNAs não implicou em alterações nos resultados. Sendo assim, o presente estudo pode ser comparado com trabalhos que utilizaram outras ferramentas para a programação das RNAs.

Nos Experimento 2 e 3, foi encontrado que é possível simular o comportamento de formação de classes de equivalência usando o procedimento *go/no-go* com estímulos compostos, mas que, para isso, é preciso haver um treino de classe adicional. Esses resultados vão ao encontro de trabalhos que simularam o comportamento de formação de classes de equivalência usando outros procedimentos e mostraram a necessidade do treino de classes adicionais (e.g. Barnes & Hampson, 1993; Lyddy & Barnes-Holmes, 2007; Tovar & Torres, 2012). É discutido que o treino de classes adicionais, já destacado como importante para o estabelecimento de classes de equivalência em animais não humanos (e.g., Schusterman & Kastak, 1993), constitue-se como um análogo à experiência pré-experimental de participantes humanos. Além disso, considerando o numero de iterações durante a fase de treino, o critério de mudança de fase foi atingido mais rapidamente quando houve treino de classe adicional, indicando que esse treino de classe adicional parece facilitar o estabelecimento de relações diretamente treinadas.

No Experimento 4, foi demonstrado que o treino de uma classe adicional contendo três elementos permite simular o comportamento de formação de classes de equivalência contendo quatro termos com o procedimento *go/no-go* com estímulos compostos. Isso indica que o treino de classes adicionais não precisa conter o mesmo número de elementos e relações presentes nas etapas subsequentes da simulação. A partir da simulação envolvendo classes contendo quatro termos, novos aspectos do comportamento de formação de classes de equivalência podem ser investigados usando-se modelos computacionais, por exemplo, o efeito de distância nodal (e.g., Doran & Fields, 2012; Fields, Adams, Verhave & Newman, 1990; Fields, Landon-Jimenez, Buffington & Adams, 1995).

Em conjunto, os resultados dos experimentos mostram que o modelo computacional desenvolvido, usando-se RNAs, é capaz de simular o comportamento de formação de classes

de equivalência pelo procedimento *go/no-go* com estímulos compostos. Por não diferenciar as funções de estímulo modelo e de estímulo de comparação, não foi preciso utilizar um recurso computacional semelhante ao duplicador do marcador do modelo das redes derivadas do RELNET, como o empregado por Barnes e Hampson, 1993. Portanto, o modelo proposto é uma alternativa ao RELNET para o estudo de relações de equivalência usando-se simulações computacionais.

Durante a simulação das tentativas de treino de relações condicionais, o algoritmo *backward propagation* foi utilizado para alterar os pesos das conexões entre as unidades de ativação. Já durante a simulação das tentativas de teste, o algoritmo foi desativado e os pesos das conexões entre as unidades de ativação foram mantidos constantes. Esse é o procedimento comumente adotado em simulações do comportamento de formação de classes de equivalência (e.g., Barnes & Hampson, 1993; Cullinan, Barnes, Hampson & Lyddy, 1994; Lyddy & Barnes-Holmes, 2007; Lyddy, Barnes-Holmes, & Hampson, 2001; Okada, Sakagami & Yamakawa, 2005; Tovar & Torres, 2012). Contudo, o fato de que os pesos das conexões são mantidos constantes durante a fase de teste faz com que não haja alteração nos vetores de saída gerados. Os desempenhos simulados são os mesmos independente de características do teste como, por exemplo, a ordem com que as relações são testadas ou a quantitade de retestes realizados. Por isso, o modelo desenvolvido não é capaz de reproduzir resultados obtidos em experimentos envolvendo humanos nos quais a ordem do teste afeta a emergência de classes de equivalência (e.g., Adams, Fields & Verhave, 1993), nem resultados em que a emergência de classes de equivalência acontece na reapresentação do teste, sem etapas adicionais de treino (e.g., Debert, Matos & McIlvane, 2007; Perez, Campos & Debert, 2009). Outra limitação do modelo desenvolvido é o fato de que a representação das tentativas de treino e teste não considerou a representação espacial dos estímulos, assim como em Tovar e Torres (2012). Por isso, não houve testes de relações de simetria, dado que os vetores de entrada de uma relação e de sua relação simétrica eram os mesmos. Tais testes sao conduzidos na maior parte dos estudos com o procedimento *go/no-go* com estímulos compostos (e.g., Debert, Matos & McIlvane, 2007; Perez, Campos & Debert, 2009; Debert, Huziwara, Faggiani, De Mathis & Mcilvane, 2009; Campos, Debert, Barros & McIlvane, 2011; Grisante et al., 2013).

Sugere-se que futuros trabalhos avaliem a possibilidade de simular as tentativas de treino e teste de relações condicionais considerando a localização espacial dos estímulos, sem utilizar recursos como o marcador do duplicador do modelo, característico das redes derivadas do RELNET. A simulação da localização espacial dos estímulos tornaria a simulação dos

desempenhos no procedimento go/no-go com estímulos compostos mais similar ao que tem sido produzido com humanos como, por exemplo, conduzir testes de simetria (Debert, Matos & McIlvane, 2007; Perez, Campos & Debert, 2009; Debert, Huziwara, Faggiani, De Mathis & Mcilvane, 2009; Campos, Debert, Barros & McIlvane, 2011; Grisante et al., 2013). Sugere-se também que futuros trabalhos investiguem a possibilidade de se utilizar um algoritmo ou outro recurso computacional durante a simulação das tentativas de teste que promovesse variabilidade dos resultados gerados nessa fase. Se houvesse variabilidade na fase de teste, seria possível simular resultados obtidos em humanos quanto aos diferentes desempenhos de acordo com a ordem das relações testadas e a quantidadede de retestes, por exemplo. Assim como feito em Lyddy e Barnes-Holmes (2007) para o procedimento MTS, seria interessante avaliar o efeito de diferentes protocolos de treino (linear, *one-to-many*, *many-to-one*) sobre os resultados das simulações do comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos, comparando-se com os resultados obtidos em experimentos com humanos (Grisante et al., 2013). Sugere-se também avaliar o efeito de distância nodal sobre os resultados das simulações computacionais e compará-los com os resultados obtidos em experimentos com humanos (e.g., Vernucio & Debert, Submetido). Sugere-se, por fim, que futuros estudos avaliem a possibilidade de adaptar o modelo desenvolvido no presente trabalho para simular comportamentos envolvendo controle condicional de segunda ordem para poder comparar com os estudos que simularam o controle contextual com o MTS (e.g., Barnes & Hampson, 1993).

Os objetivos de uma ciência do comportamento são a previsão e o controle do comportamento (Watson, 1913). Para a previsão, simulações computacionais são ferramentas essenciais. Contudo, os modelos computacionais existentes atualmente ainda possuem limitações. É preciso aumentar a complexidade desses modelos para que previsões cada vez mais precisas sejam feitas, possibilitando que simulações computacionais sejam integradas a experimentos envolvendo seres vivos para gerar novos conhecimentos.

REFERÊNCIAS[2]

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. (1993). Stimulus equivalence and connectionism: Implications for behavior analysis and cognitive science. *The Psychological Record, 43*, 617-638.

Bickel, K. W., Richmond, G., Bell, J., & Brown, K, (1986). A microanalysis of the controlling stimulus-response relations engendered during the assessment of stimulus overselectivity. *The Psychological Record, 36*, 225-238.

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.

Campos, H. C., Debert, P., Barros, R. S., & McIlvane, W. J. (2011). Relational Discrimination by Pigeons in a Go/No-go Procedure With Compound Stimuli: A Methodological Note. *Journal of the Experimental Analysis of Behavior, 96*, 417-426.

Debert, P., Huziwara, E. M., Faggiani, R. B., De Mathis, M. E. S., Mcilvane, W. J. (2009). Emergent Conditional Relations in a Go/No-Go Procedure: Figure ground and Stimulus-Position Compound Relations. *Journal of the Experimental Analysis of Behavior, 92*, 233-243.

Debert, P., Matos, M. A., & McIlvane, W. J. (2007). Conditional Relations with Compound Abstract Stimuli Using a Go/No-Go Procedure. *Journal of the Experimental Analysis of Behavior, 87*, 89-96.

Doran, E., & Fields, L. (2012). All stimuli are equal, but some are more equal than others: measuring relational preferences within an equivalence class. *Journal of the Experimental Analysis of Behavior, 98*(3), 243–256.

Fields, L., Adams, B. J., Verhave, T., & Newman, S. (1990). The effects of nodality on the formation of equivalence classes. *Journal of the Experimental Analysis of behavior, 53*(3), 345-358.

[2] De acordo com o estilo APA – *American Psycological Association*.

Fields, L., Doran, E., & Marroquin, M. (2009). Equivalence class formation in a trace stimulus pairing two-response format: effects of response labels and prior programmed transitivity induction. *Journal of the Experimental Analysis of Behavior*, *92*(1), 57-84.

Fields, L., Landon-Jimenez, D. V., Buffington, D. M., & Adams, B. J. (1995). Maintained nodal-distance effects in equivalence classes. *Journal of the Experimental Analysis of behavior*, *64*(2), 129-145.

Fields, L., Reeve, K. F., Varelas, A., Rosen, D., & Belanich, J. (1997). Equivalence class formation using stimulus-paring and yes-no responding. *The Psychological Record, 47*, 661-686.

Grisante, P. C., Galesi, F. L., Sabino, N. M., Debert, P., Arntzen, E., & McIlvane, W. J . (2013). Go/no-go procedure with compound stimuli: Effects Of Training Structure On The Emergence Of Equivalence Classes. *The Psychological Record, 63*, 63-72.

Haykin, S. (2001). *RNAs Neurais: Princípios e prática*. Porto Alegre: Bookman

Hayes, S. C., & Hayes, L. J. (1989). The verbal action of the listener as a basis for rule-governance. In *Rule-governed behavior: Cognition, contingencies, and instructional control*, 153-190. New York: Plenum.

Kemp, S. M., & Eckerman, D. A. (2001). Situational descriptions of behavioral procedures: The In Situ testbed. *Journal of the experimental analysis of behavior*, *75*(2), 135-164.

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.

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, v. 1.4-2.1*, 14-24.

McClelland, J. L. (2009). The place of modeling in cognitive science. *Topics in Cognitive Science, 1*(1), 11-38.

McElreath, R., & Boyd, R. (2008). *Mathematical models of social evolution: a guide for the perplexed*. University of Chicago Press.

Nissen, S. (2003). Implementation of a fast artificial neural network library (fann). *Report, Department of Computer Science University of Copenhagen (DIKU), 31*.

Okada, H., Sakagami, M., & Yamakawa, H. (2005). Modeling Stimulus Equivalence with Multi Layered Neural Networks. Computational Intelligence and Bioinspired Systems, pp 153-160. *Lecture Notes in Computer Science,3512*, Springer Berlin Heidelberg.

Perez, W., Campos, H. C., & Debert, P. (2009). Procedimento go/no-go com estímulos compostos e a emergência de duas classes com três estímulos. *Acta Comportamentalia, 17*, 191-210.

Plunkett, K., & Elman, J. L. (1997*). Exercises in rethinking innateness: A handbook for connectionist simulations*. MIT Press.

Ritchie, D. M., Kernighan, B. W., & Lesk, M. E. (1975). *The C programming language*. Bell Laboratories.

Rumelhart, D. E., Hinton, G. E., & Williams, D. C. (1986). Learning representations by back-propagating errors. *Nature, 323,* 533-536.

Schusterman, R. J., & Kastak, D. (1993). A California sea lion (Zalophus californianus) is capable of forming equivalence relations. *Psychological Record, 43*(4), 823-839.

Sidman, M., & Tailby, W. (1982). Conditional discrimination vs. matching to sample: An expansion of the testing paradigm. *Journal of the Experimental Analysis of Behavior, 53,* 47-63.

Sidman, M. (1994). *Equivalence relations and behavior: A research story*. Boston: Authors Cooperative Pub.

Skinner, B. F. (1953). *Science and human behavior*. New York: Macmillan.

Steele, D. L., & Hayes, S. C. (1991). Stimulus equivalence and arbitrarily applicable

relational responding. *Journal of the Experimental Analysis of Behavior, 56*, 519-555.

Tovar, A. E., & Torres, A. (2012). A connectionist model of stimulus class formation with a yes-no procedure and compound stimuli. *The Psychological Record, 62*, 747-762.

APÊNDICE A – Código fonte do modelo computacional usado para a replicação sistemática de Tovar e Torres (2012)

```
/*
Rede Neural Artificial - Yes-no
Autor: Renato Vernucio - renato.vernucio@usp.br
Data: 06/2014

Cria uma rede neural artificial feed-forward totalmente conectada, com
neuronios sigmoidais, para ser treinada utilizando-se backpropagation. A
rede representa um modelo de formacao de classes de equivalencia por meio
do precidemento Yes-no com estimulos compostos. Replicacao de Tovar e
Torres (2012).
*/

#include <stdio.h>
#include <time.h>
#include "fann.h"

int main()
{
    /* variaveis de treino */
    const unsigned int num_layers = 3;
    const unsigned int num_input = 9;
    const unsigned int num_neurons_hidden = 4;
    const unsigned int num_output = 2;
    const float weight_min = (const float) 0.001;
    const float weight_max = (const float) 1.0;
    const float mse_desired = (const float) 0.0025;
    const unsigned int epochs_max = 500000;
    unsigned int epochs_current = 0;
    const unsigned int epochs_between_reports = 0;
    float mse_current;

    /* variaveis de teste */
    unsigned short int i = 0;
    struct fann_train_data *dataset_train =
        fann_read_train_from_file("yesno.data");
    struct fann_train_data *dataset_test =
        fann_read_train_from_file("yesno.test.data");
    fann_type input_train[11][9] = {
                                     {1, 0, 0, 0, 0, 0, 0, 1, 0},
                                     {1, 0, 1, 0, 0, 0, 0, 0, 0},
                                     {0, 0, 0, 0, 1, 0, 0, 1, 0},
                                     {0, 0, 0, 0, 0, 0, 1, 1, 0},
                                     {0, 0, 1, 0, 0, 0, 0, 0, 1},
                                     {0, 0, 0, 0, 0, 0, 0, 1, 1},
                                     {0, 0, 1, 0, 0, 0, 1, 0, 0},
                                     {0, 0, 1, 0, 1, 0, 0, 0, 0},
                                     {0, 0, 0, 1, 0, 1, 0, 0, 0},
                                     {0, 1, 0, 0, 0, 1, 0, 0, 0},
                                     {0, 1, 0, 1, 0, 0, 0, 0, 0}
                                    };
    fann_type input_test[4][9] = {
                                     {1, 0, 0, 0, 1, 0, 0, 0, 0},
                                     {1, 0, 0, 0, 0, 0, 1, 0, 0},
                                     {0, 0, 0, 0, 0, 0, 1, 0, 1},
```

```
                                    {0, 0, 0, 0, 1, 0, 0, 0, 1}
                                };
    fann_type *output;


    /* variaveis de registro de execucao */
    FILE *fp;
    time_t date_time;
    char *date_time_str;

    date_time = time(NULL);
    date_time_str = ctime(&date_time);
    fp = fopen ("log.txt", "a");
    fprintf(fp, date_time_str);

    /* cria rede neural */
    struct fann *ann = fann_create_standard(num_layers, num_input,
        num_neurons_hidden, num_output);
    fann_set_activation_function_hidden(ann, FANN_SIGMOID);
    fann_set_activation_function_output(ann, FANN_SIGMOID);
    fann_randomize_weights(ann, weight_min, weight_max);

    /* define parametros de treino */
    fann_set_train_stop_function(ann, FANN_STOPFUNC_MSE);
    fann_set_training_algorithm(ann, FANN_TRAIN_INCREMENTAL); /*_BATCH*/
    fann_set_learning_rate(ann, 0.3);
    fann_set_learning_momentum(ann, 0.0);

    /* executa treino AB e BC ate LMS < mse_desired */
    do
    {
        fann_shuffle_train_data(dataset_train);
        mse_current = fann_train_epoch(ann, dataset_train); /* mse
impreciso */
        mse_current = fann_test_data(ann, dataset_train);
        epochs_current++;
    } while (mse_current >= mse_desired);
    fprintf(fp, "Treino finalizado em %u epochs, com MSE = %1.5f\n",
        epochs_current, mse_current);


    /* testa relacoes treinadas */
    fprintf(fp, "Teste de relacoes treinadas (MSE = %1.5f):\n",
        mse_current);
    for(i = 0; i <= 10; i++)
    {
        output = fann_run(ann, input_train[i]);
        fprintf(fp, "(%1.0f,%1.0f,%1.0f,%1.0f,%1.0f,%1.0f,%1.0f,"
            "%1.0f,%1.0f) -> (%1.5f , %1.5f)\n", input_train[i][0],
            input_train[i][1], input_train[i][2], input_train[i][3],
            input_train[i][4], input_train[i][5], input_train[i][6],
            input_train[i][7], input_train[i][8], output[0], output[1]);
    }

    /* testa AC */
    fprintf(fp, "Teste de relacoes emergentes (MSE = %1.5f):\n",
        fann_test_data(ann, dataset_test));
    for(i = 0; i <= 3; i++)
    {
        output = fann_run(ann, input_test[i]);
        fprintf(fp, "(%1.0f,%1.0f,%1.0f,%1.0f,%1.0f,%1.0f,%1.0f,"
```

```
            "%1.0f,%1.0f) -> (%1.5f , %1.5f)\n", input_test[i][0],
            input_test[i][1], input_test[i][2], input_test[i][3],
            input_test[i][4], input_test[i][5], input_test[i][6],
            input_test[i][7], input_test[i][8], output[0], output[1]);
    }

    fprintf(fp, "\n\n");
    fann_destroy(ann);
    fclose(fp);

    return 0;
}
```

APÊNDICE B – Código fonte do modelo computacional usado para simular o comportamento de formação de classes de equivalência pelo procedimento *go/no-go* com estímulos compostos.

```
/*
Rede Neural Artificial - Procedimento Go/No-go
Autor: Renato Vernucio - renato.vernucio@usp.br
Data: 04/2015

Cria uma rede neural artificial do tipo feedforward totalmente conectada e
com funcao de ativacao sigmoidal. O treino eh realizado utilizando-se o
algoritmo backpropagation. A rede apresenta-se como um modelo conexionista
para simular o comportamento de formacao de classes de equivalencia por
meio do procedimento Go/No-go com estimulos compostos. O codigo foi criado
para executar os Experimentos 2, 3 e 4 da pesquisa de mestrado.

O modelo computacional desenvolvido eh generico e pode ser utilizado, sem
alteracao do codigo fonte, para simular quaisquer relacoes de treino e
teste utilizando-se o procedimento Go/No-go. Para configurar os diferentes
datasets de treino e teste, deve-se alterar os arquivos train.data e
test.data conforme um formato especifico. Na primeira linha devem ser
informados, separados por um único espaco, o numero de relacoes, o tamanho
do input e o tamanho do output, respectivamente. Nas demais linhas, deve-se
intercalar um vetor de entrada e, na linha subsequente, seu vetor de saida
correspondente:

numero_de_relacoes[linhas] tamanho_do_input[colunas]
tamanho_do_output[colunas]
[vetor de entrada 1, com valores separados por espacos]
[vetor de saida 1 correspondente, com valores separados por espacos]
[vetor de entrada 2, com valores separados por espacos]
[vetor de saida 2 correspondente, com valores separados por espacos]
(...)

Para mais informacoes sobre o formato dos arquivos *,data, consultar a
documentacao da FAAN. Se for necessario alterar os parametros do treino,
como quantidade de camadas, quantidade de unidades na camada intermediaria,
valor do MSE como criterio de parada etc, deve-se alterar apenas as
diretivas de pre-processamento #define do inicio do codigo, mantendo-se
todo o resto inalterado.
*/

#include <stdio.h>
#include <time.h>
#include <stdbool.h>
#include <unistd.h>
#include <fann.h>
#include <floatfann.c>

#define NUM_LAYERS          3
#define NUM_NEURONS_HIDDEN  4
#define MSE                 0.0025
#define MAX_ERROR           0.15
#define LEARNING_RATE       0.3
#define LEARNING_MOMENTUM   0.0
#define TRAIN_FILE          "train.data"
```

```
#define TEST_FILE           "test.data"
#define LOG_FILE            "log.csv"
#define DEBUG_FILE          "debug_log.txt"
#define CSV_DELIMITER       ","
#define CSV_EXPORT_TO_EXCEL true
#define DEBUG               false


int file_already_exists(const char *file);

int main()
{
    /* variaveis de treino e teste */
    unsigned int num_input;
    unsigned int num_output;
    unsigned int num_relations_train;
    unsigned int num_relations_test;
    const unsigned int num_layers = NUM_LAYERS;
    const unsigned int num_neurons_hidden = NUM_NEURONS_HIDDEN;
    const float learning_rate = LEARNING_RATE;
    const float learning_momentum = LEARNING_MOMENTUM;
    const float max_error = MAX_ERROR;
    const float weight_min = (const float) 0.0;
    const float weight_max = (const float) 1.0;
    const float mse_desired = (const float) MSE;
    float mse_current;
    float mse_train_final;
    float mse_test_final;
    unsigned int epochs_current = 0;
    unsigned int epochs_total;
    bool successful_training = true;
    bool successful_testing = true;

    /* variaveis de manipulacao de arquivos */
    bool debug = DEBUG;
    const char *debug_file = DEBUG_FILE;
    const char *log_file = LOG_FILE;
    const char *train_file = TRAIN_FILE;
    const char *test_file = TEST_FILE;
    FILE *f_debug;
    FILE *f_log;
    FILE *f_train;
    FILE *f_test;
    char *csv_delimiter = CSV_DELIMITER;
    bool csv_export_to_excel = CSV_EXPORT_TO_EXCEL;

    /* estruturas para dataset de treino e teste */
    struct fann_train_data *dataset_train =
        fann_read_train_from_file(train_file);
    struct fann_train_data *dataset_test =
        fann_read_train_from_file(test_file);

    /* variaveis de loop */
    unsigned short int i;
    unsigned short int j;
    unsigned short int input_i;
    unsigned short int output_i;

    /* vetores de entrada e saida da fase de treino de relacoes diretas */
    f_train = fopen(train_file, "r");
    if(f_train == NULL)
```

```
    {
        printf("ERRO: impossivel abrir %s. Simulacao abortada.\n",
train_file);
        return 1;
    }
    fscanf(f_train, "%u %u %u\n", &num_relations_train, &num_input,
        &num_output);
    fann_type **input_train =
        malloc(num_relations_train * sizeof(fann_type *));
    for(i = 0; i <= num_relations_train - 1; i++)
        input_train[i] = malloc(num_input * sizeof(fann_type));
    float *output_train_desired =
        malloc(num_relations_train * sizeof(float));
    float *output_train_received =
        malloc(num_relations_train * sizeof(float));
    float *output_train_error =
        malloc(num_relations_train * sizeof(float));
    input_i = 0;
    output_i = 0;
    for(i = 0; i <= (num_relations_train * 2) - 1; i++)
    {
        if(i % 2 == 0)  /* linha par, que contem vetor de entrada */
        {
            for(j = 0; j <= num_input - 2; j++)
                fscanf(f_train, "%f ", &input_train[input_i][j]);
            fscanf(f_train, "%f\n", &input_train[input_i][j]);
            input_i++;
        }
        else  /* linha impar, que contem vetor de saida */
        {
            fscanf(f_train, "%f\n", &output_train_desired[output_i]);
            output_i++;
        }
    }
    fclose(f_train);

    /* vetores de entrada e saida da fase de teste de emergencia de
relacoes */
    f_test = fopen(test_file, "r");
    if(f_test == NULL)
    {
        printf("ERRO: impossivel abrir %s. Simulacao abortada.\n",
test_file);
        return 1;
    }
    fscanf(f_test, "%u %u %u\n", &num_relations_test, &num_input,
&num_output);
    fann_type **input_test =
        malloc(num_relations_test * sizeof(fann_type *));
    for(i = 0; i <= num_relations_test - 1; i++)
        input_test[i] = malloc(num_input * sizeof(fann_type));
    float *output_test_desired =
        malloc(num_relations_test * sizeof(float));
    float *output_test_received =
        malloc(num_relations_test * sizeof(float));
    float *output_test_error =
        malloc(num_relations_test * sizeof(float));
    input_i = 0;
    output_i = 0;
    for(i = 0; i <= (num_relations_test * 2) - 1; i++)
    {
```

```
        if(i % 2 == 0)  /* linha par, que contem vetor de entrada */
        {
            for(j = 0; j <= num_input - 2; j++)
                fscanf(f_test, "%f ", &input_test[input_i][j]);
            fscanf(f_test, "%f\n", &input_test[input_i][j]);
            input_i++;
        }
        else  /* linha impar, que contem vetor de saida */
        {
            fscanf(f_test, "%f\n", &output_test_desired[output_i]);
            output_i++;
        }
    }
    fclose(f_test);
    fann_type *output_curr;

    /* variaveis de registro de execucao */
    time_t date_time;
    char *date_time_str;

    /* registro da data, hora, minuto e segundo da execucao */
    date_time = time(NULL);
    date_time_str = ctime(&date_time);
    strtok(date_time_str, "\n");

    /* cria rede neural artificial para simulacao */
    struct fann *ann = fann_create_standard(num_layers, num_input,
        num_neurons_hidden, num_output);
    fann_set_activation_function_hidden(ann, FANN_SIGMOID);
    fann_set_activation_function_output(ann, FANN_SIGMOID);
    fann_randomize_weights(ann, weight_min, weight_max);

    /* define parametros de treino */
    fann_set_train_stop_function(ann, FANN_STOPFUNC_MSE);
    fann_set_training_algorithm(ann, FANN_TRAIN_INCREMENTAL); /*_BATCH*/
    fann_set_learning_rate(ann, learning_rate);
    fann_set_learning_momentum(ann, learning_momentum);

    /* treina relacoes diretas ate criterio de parada MSE ser atingido */
    do
    {
        fann_shuffle_train_data(dataset_train);
        mse_current = fann_train_epoch(ann, dataset_train); /* mse
impreciso */
        mse_current = fann_test_data(ann, dataset_train);   /* mse preciso
*/
        epochs_current++;
    } while (mse_current >= mse_desired);
    mse_train_final = mse_current;
    epochs_total = epochs_current;

    /* testa relacoes diretas treinadas */
    for(i = 0; i <= num_relations_train - 1; i++)
    {
        output_curr = fann_run(ann, input_train[i]);
        output_train_received[i] = output_curr[0];
        output_train_error[i] = fabsf(
            output_train_desired[i] - output_train_received[i]);
        if(output_train_error[i] > max_error)
            successful_training = false;
    }
```

```
    /* testa relacoes emergentes */
    for(i = 0; i <= num_relations_test - 1; i++)
    {
        output_curr = fann_run(ann, input_test[i]);
        output_test_received[i] = output_curr[0];
        output_test_error[i] = fabsf(
            output_test_received[i] - output_test_desired[i]);
        if(output_test_error[i] > max_error)
            successful_testing = false;
    }
    mse_test_final = fann_test_data(ann, dataset_test);

    /* finaliza simulacao */
    fann_destroy(ann);

    /* registro em arquivo e exibicao em stdout para debug da simulacao */
    if(debug)
    {
        /* imprime em stdout informacoes sobre vetores de entrada e saida
*/
        fprintf(stdout, "input_train\n");
        for(i = 0; i <= num_relations_train - 1; i++)
        {
            for(j = 0; j <= num_input - 1; j++)
                fprintf(stdout, "%1.0f ", input_train[i][j]);
            fprintf(stdout, "\n");
        }
        fprintf(stdout, "\n\n");

        fprintf(stdout, "input_test\n");
        for(i = 0; i <= num_relations_test - 1; i++)
        {
            for(j = 0; j <= num_input - 1; j++)
                fprintf(stdout, "%1.0f ", input_test[i][j]);
            fprintf(stdout, "\n");
        }
        fprintf(stdout, "\n\n");

        fprintf(stdout, "output_train_desired\n");
        for(i = 0; i <= num_relations_train - 1; i++)
            fprintf(stdout, "%1.0f\n", output_train_desired[i]);
        fprintf(stdout, "\n\n");

        fprintf(stdout, "output_test_desired\n");
        for(i = 0; i <= num_relations_test - 1; i++)
            fprintf(stdout, "%1.0f\n", output_test_desired[i]);
        fprintf(stdout, "\n\n");

        /* inicializa o arquivo de log das simulacoes */
        if(file_already_exists(debug_file))
            f_debug = fopen(debug_file, "a");
        else
            f_debug = fopen(debug_file, "w");

        /* cabecalho de referencia */
        fprintf(f_debug, "%s\n", date_time_str);
        fprintf(f_debug, "Treino finalizado em %u epochs, com MSE
= %1.5f\n",
            epochs_total, mse_train_final);
```

```
        /* vetores de saida para relacoes treinadas */
        /* imprime (vetor de entrada) -> (vetor de saida) */
        fprintf(f_debug, "Teste de relacoes treinadas (MSE = %1.5f):\n",
            mse_train_final);
            for(i = 0; i <= num_relations_train - 1; i++)
            {
                fprintf(f_debug, "(");
                for(j = 0; j <= num_input - 2; j++)
                    fprintf(f_debug,"%1.0f,", input_train[i][j]);
                fprintf(f_debug,"%1.0f)", input_train[i][j]);
                fprintf(f_debug, " -> (%1.5f)", output_train_received[i]);
                if(output_train_error[i] <= max_error)
                    fprintf(f_debug, " [OK]\n");
                else
                    fprintf(f_debug, " [ERRO]\n");
            }

        /* vetores de saida para relacoes emergentes */
        /* imprime (vetor de entrada) -> (vetor de saida) */
        fprintf(f_debug, "Teste de relacoes emergentes (MSE = %1.5f):\n",
            mse_test_final);
        for(i = 0; i <= num_relations_test - 1; i++)
        {
            fprintf(f_debug, "(");
            for(j = 0; j <= num_input - 2; j++)
                fprintf(f_debug,"%1.0f,", input_test[i][j]);
            fprintf(f_debug,"%1.0f)", input_test[i][j]);
            fprintf(f_debug, " -> (%1.5f)", output_test_received[i]);
            if(output_test_error[i] <= max_error)
                fprintf(f_debug, " [OK]\n");
            else
                fprintf(f_debug, " [ERRO]\n");
        }
        fprintf(f_debug, "\n\n");
        fclose(f_debug);
    }
    /* registro da simulacao em formato CSV */
    else
    {
        /* abre arquivo pre-existente para log das simulacoes, se existir
*/
        if(file_already_exists(log_file))
        {
            f_log = fopen(log_file, "a");
            if(f_log == NULL)
            {
                printf("ERRO: impossivel abrir %s. Simulacao abortada.\n",
                    log_file);
                return 1;
            }
        }
        /* inicializa um novo arquivo e cria o cabecalho para log */
        else
        {
            f_log = fopen(log_file, "w");
            if(f_log == NULL)
            {
                printf("ERRO: impossivel abrir %s. Simulacao abortada.\n",
                    log_file);
                return 1;
            }
```

```
            if(csv_export_to_excel)
                fprintf(f_log, "sep=,\n");
            /* cria cabecalho do arquivo CSV */
            fprintf(f_log, "DATA%sEPOCHS%sMSE TREINO%sPERF. TREINO%s",
                csv_delimiter, csv_delimiter, csv_delimiter, csv_delimiter);
            for(i = 0; i <= num_relations_train - 1; i++)
            {
                fprintf(f_log, "\"(");
                for(j = 0; j <= num_input - 2; j++)
                    fprintf(f_log, "%1.0f%s", input_train[i][j],
                        csv_delimiter);
                fprintf(f_log, "%1.0f", input_train[i][j]);
                fprintf(f_log, ")\"%s", csv_delimiter);
            }
            fprintf(f_log, "MSE TESTE%sPERF. TESTE%s", csv_delimiter,
                csv_delimiter);
            for(i = 0; i <= num_relations_test - 1; i++)
            {
                fprintf(f_log, "\"(");
                for(j = 0; j <= num_input - 2; j++)
                    fprintf(f_log, "%1.0f%s", input_test[i][j],
                        csv_delimiter);
                fprintf(f_log, "%1.0f", input_test[i][j]);
                if(i < num_relations_test)
                    fprintf(f_log, ")\"%s", csv_delimiter);
                else
                    fprintf(f_log, ")\"");
            }
            fprintf(f_log, "\n");
        }

    /* imprime no arquivo de log uma linha contendo dados da simulacao:
    DATA | EPOCHS | MSE TREINO | PERF. TREINO | [VETORES DE ENTRADA] |
    MSE TESTE | PERF. TESTE | [VETORES DE SAÍDA]
    */
    fprintf(f_log, "%s%s%u%s%1.5f%s", date_time_str, csv_delimiter,
        epochs_total, csv_delimiter, mse_train_final, csv_delimiter);
    if(successful_training)
        fprintf(f_log, "OK%s", csv_delimiter);
    else
        fprintf(f_log, "ERRO%s", csv_delimiter);
    /* vetores de saida na fase de treino */
    for(i = 0; i <= num_relations_train - 1; i++)
        fprintf(f_log, "%1.5f%s", output_train_received[i], csv_delimiter);
    fprintf(f_log, "%1.5f%s", mse_test_final, csv_delimiter);
    if(successful_testing)
        fprintf(f_log, "OK%s", csv_delimiter);
    else
        fprintf(f_log, "ERRO%s", csv_delimiter);

    /* vetores de saida na fase de teste */
    for(i = 0; i <= num_relations_test - 1; i++)
        fprintf(f_log, "%1.5f%s", output_test_received[i], csv_delimiter);
    fprintf(f_log, "\n");
    }

    return 0;
}

/* verifica se um arquivo ja existe */
int file_already_exists(const char *file)
```

```
{
    if(access(file, F_OK) != -1)
        return true;
    else
        return false;
}
```