# Teste e Implementação de uma rede neural utilizando redes feedfoward aplicadas em BCPs

**Andrezza de M. Bonfim**[1], **Edson S. do N. Neto**[1], **Nader M. Hauache.**[1]

[1]Universidade Federal do Amazonas (UFAM)

andrezzabonfiiim@gmail.com, netosilvan78@gmail.com, naderhauache@gmail.com

***Abstract.*** *In this paper, we intend to perform a test of a proposal made by França, et al[1], aiming at the implementation and execution of a neural network using CILP++ method for relational learning. Relational learning is defined as the task of learning logic rules from propositional examples. There are systems called "ILP Systems" (Inductive Logic Programming Systems) whose main goal is expression, through propositional rules, of first-order logic. CILP++ method aims, through BCPs(Bottom Clause Propositionalization) - structures used on ILP Systems - to learn first-order logic from examples, performing the so-called "Neuro-Symbolic Learning". But, Instead of backpropagation-based neural nets, we used FeedFoward ones.*

***Resumo.*** *Neste artigo, buscamos realizar um teste da proposta feita por França, et al[1], objetivando a implementação e execução de uma rede neural que utiliza-se do método CILP++ para aprendizado relacional. Aprendizado relacional é definido como a tarefa de aprender regras lógicas a partir de exemplos proposicionais. Existem sistemas chamados "sistemas ILP" (Inductive Logic Programming) onde seu principal objetivo é expressar, através de regras proposicionais, lógica de primeira ordem. O método CILP++[1] busca, através de BCPs(Bottom Clause Propositionalization) - estruturas usadas em sistemas ILP, que carregam o significado lógico e podem ser transcritas em vetores númericos - aprender regras de lógica de primeira ordem através de exemplos, realizando assim o chamado "aprendizado neuro-simbólico". Mas no lugar de aplicarmos o método "backpropagation" nas conexões neurais, fora utilizado o método de otimização de pesos "FeedFoward".*

## 1. Introdução

O objetivo humano de ensinar uma máquina a reconhecer padrões e aprender com eles através de aprendizagem de máquina é explicado por Russel e Norvig, no cap.18[2]:

> [...]Há três razões principais. Primeiro, os projetistas não podem antecipar todas as situações possíveis em que o agente possa se encontrar. Por exemplo, um robô projetado para navegar em labirintos tem de aprender a configuração de cada novo labirinto que encontra. Em segundo lugar, os projetistas não podem antecipar todas as mudanças ao longo do tempo; um programa projetado para prever os preços do mercado de ações de amanhã deve aprender a se adaptar quando as condições mudam do súbito crescimento ao fracasso. Terceiro, por vezes, os programadores humanos não têm ideia de como programar uma solução por si só. Por exemplo, a

> maioria das pessoas é boa em reconhecer o rosto dos membros da família, mas mesmo os melhores programadores são incapazes de programar um computador para realizar essa tarefa, exceto por meio de algoritmos de aprendizagem.[...]

Dentro disso, ao longo dos anos, vários foram os avanços nesssa área de Inteligência Artificial, sendo os principais tópicos caracterizados de acordo com a tabela da Fig.1.

| Modelo de Aprendizado | Paradigmas de Aprendizado | Linguagens de Descrição | Formas de Aprendizado |
|---|---|---|---|
| Supervisionado | Simbólico | Instâncias ou Exemplos | Incremental |
| Não-Supervisionado | Estático | Conceitos Aprendidos ou Hipóteses | Não Incremental |
| | Instance-Based | Teoria de Domínio ou Conhecimento de Fundo | |
| | Conexionista | | |
| | Genético | | |

**Figure 1. Características do aprendizado de máquinas**

### 1.1. Modelos de Aprendizado

No aprendizado supervisionado, o agente ”recebe” os dados de Input e qual deve ser o Output para cada um deles, e com isso, reconhece padrões, ”aprendendo” assim uma função que faça o mapeamento de entrada e saída. E no não supervisionado, o agente recebe padrões de entrada sem receber nenhum ”feedback” explícito; com isso, passa a tentar ”agrupar” os dados de entrada com suas respectivas saídas, tentando identificar um padrão de agrupamento/classificação.

### 1.2. Paradigmas de Apredizado

Dentre os modelos Citados na Fig.1, o Simbólico talvez seja o mais simples de todos, pois utiliza de regras lógicas, árvores de decisão e simbolismo formal para realizar inferências lógicas diversas. O Estatístico, em contra partida, busca utilizar de métodos de inferência estatística para inferir conclusões sobre determinado tipos de dados. O Induced-Base Learning é um sistema baseado na análise de casos, onde busca-se a partir da análise de um ”set” pré definido, buscar as melhores soluções. O Genético, utiliza-se da teoria evolucionista para que seja definida uma solução para um problema desconhecido, e por fim, o conexionista, por sua vez, é majoritariamente composto pelas chamadas Redes Neurais onde possui-se uma quantidade significativa de neurônios que farão a classificação e a seleção natural dos dados desejados, baseado no modelo biológico do sistema nervoso.

### 1.3. Linguagens de Dedução

Quando é dito que a Linguagem de Dedução de um agente é baseada em instâncias, é que tem-se um conjunto geralmente grande, de exemplos com suas determinadas classes, onde é buscado atingir um bom classificador para tal. Quando baseado em Hipotéses, é que a partir delas, analisa-se a veracidade de validade para o resultado final esperado, e com isso, verifica-se se aquela hipótese está inserida no conjunto positivo ou negativo dentro do espaço de soluções. Por outro lado, utilizando Teoria de Domínio, a partir de um dataset já apresentado, busca-se expandir com os dados já possuidos a base de dados, inferindo novas regras.

### 1.4. Formas de Aprendizado

Enquanto a forma Incremental atualiza o conhecimento do Agente a cada novo exemplo ingressado ao sistema indutor, a Não Incremental apenas atualiza o conhecimento na base de dados quando todos os exemplos já foram apresentados.

## 2. Fundamentos de Aprendizado por Indução em Neuro-Symbolic System

O método CILP++[1] é uma evolução do CILP[2]. O CILP foi uma proposta realizada à comunidade científica de uma integração neuro simbólica para explorar as vantagens e benefícios dos paradigmas de aprendizado indutivo de redes neurais com aprendizado dedutivo de de programação lógica. O Trabalho descrito por Garcez[2] propõe que um dataset de regras FOL(First Order Logic) pode ser convertido para uma rede neural de 3 camadas - Input Layer, Hidden Layer & Output Layer - e com isso, realizar aprendizado indutivo, aproveitando o melhor nas Redes Neurais Artificiais, como paralelismo massivo, aprendizado indutivo e capacidade de generalização, junto ao melhor dos sistemas simbólicos, como a explicação so processo de inferência e do processo declarativo das linguagens de representação de conhecimento. O método CILP++, e o CILP, por padrão, utilizam-se de Backpropagation na rede neural para ajuste de pesos. Em nosso trabalho, por questões de simplificação, utilizou-se do método Feedfoward, ao invés do idealizado Backpropagation[1][2].

Na estratégia de aprendizado por indução, o sistema aprendiz adquire os conceitos através de inferências indutivas realizadas sobre fatos fornecidos ou observados. De forma geral, derivar conclusões gerais a partir de observações específicas é chamado de indução. Dessa forma, no Aprendizado Indutivo, os conceitos são de fato induzidos, uma vez que as hipóteses (conclusões) são induzidas a partir dos exemplos (observações específicas). Este processo de indução pode ser visto como um processo de busca em que se almeja encontrar a melhor hipótese (de acordo com certa função de avaliação) no espaço de todas as possíveis hipóteses (MITCHELL, 1997).

A conjectura fundamental do aprendizado indutivo é que qualquer hipótese descoberta que aproxima bem um determinado conceito (função alvo) usando um conjunto suficientemente grande de exemplos de treinamento, também aproximará bem a função alvo sobre os outros exemplos não observados (generalização) (MITCHELL, 1997).

### 2.1. Linguagens de Descrição

Uma linguagem de descrição deve ser escolhida para representar os conceitos, as hipóteses, os exemplos, e o conhecimento preliminar. A escolha da linguagem de representação é muito importante, uma vez que ela limita o tipo de conceito que pode ser aprendido. Com uma linguagem de representação detentora de uma baixa expressividade, pode-se não ser possível representar algum problema, por ele ser muito complexo para a linguagem adotada. Em contrapartida, uma linguagem com grande expressividade pode ser capaz de representar um maior conjunto de domínios de problemas. Entretanto, esta solução pode fornecer muita liberdade no sentido de ser possível construir hipóteses de muitas formas diferentes, o que poderia levar a uma impossibilidade de se encontrar o conceito correto. Como linguagem de representação, ILP (inductive logic programming) utiliza um fragmento da Lógica de Primeira Ordem.

Alguns símbolos de Lógica de Primeira Ordem conhecidos estão na Figura 2. Essa linguagem possibilita uma enorme capacidade de representação dos problemas a serem solucionados.

1. **Símbolos Lógicos:**
    a. Pontuação: consiste de parêntese aberto, parêntese fechado e vírgula.
    b. Conectivos: $\neg$ (negação), $\wedge$ (conjunção), $\vee$ (disjunção), $\leftarrow$ (implicação), e $\leftrightarrow$ (bi-condicional).
    c. Quantificadores: $\forall$ (universal), e $\exists$ (existencial).
    d. Variáveis: se iniciam com letras maiúsculas, tais como A, X, Var.
    e. Símbolo de igualdade (opcional): =
2. **Símbolos não-lógicos:**
    a. Funções: se iniciam por letras minúsculas.
    b. Constantes: são símbolos funcionais de aridade (número de argumentos) zero.
    c. Predicados: expressam relações entre seus argumentos.

**Figure 2. Símbolos de Lógica de Primeira Ordem**

Dentro disso, dada a necessidade de expressar essas regras, criaram-se diversos sistemas e linguagens de programação, como Prolog e Progol.

### 2.2. Redes Neurais

Rede neural é um modelo computacional criado para resolver problemas difícieis de resolver se utilizassemos regras comuns de representação. Baseado no Sistema Nervoso Central de um animal, são utilizadas largamente em aprendizado de máquina e reconhecimento de padrões, atividades facilitadas a partir desse modelo. Basicamente, podem ser descritas como ”neurônios interconectados que computam vários valores de entradas”, simulando redes neurais biológicas.

É uma estrutura capaz de aprender, desde que treinadas, possuem uma estrutura interna auto organizável, tem processos embarcados de tolerância a falhas, são flexíveis, pois aceitam dados muito heterogêneos e possuem respota em tempo real dependendo de sua implementação.

E para tais redes, existem alguns algoritmos que ditam comportamento de tal sistema, dentre eles o Feedfoward e o Backpropagation.

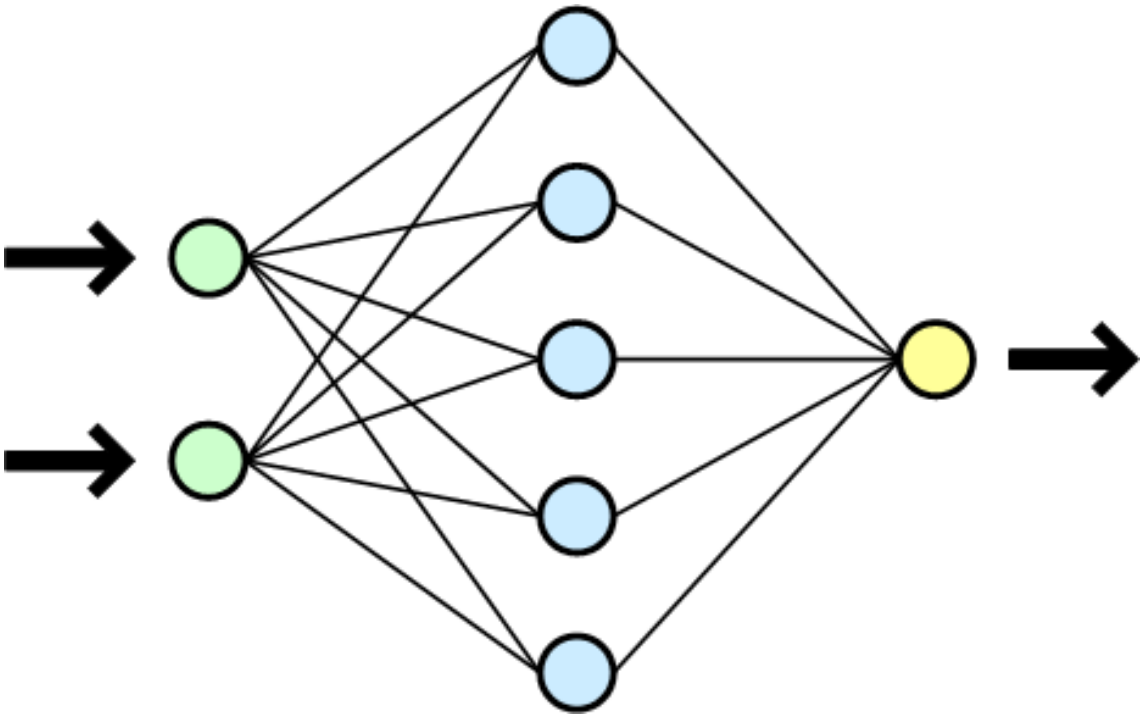

**Figure 3. Rede Neural com 3 Layers**

### 2.3. Feedfoward vs. Backpropagation

Uma rede que utilize do método FeedFoward, é uma rede neural na qual não há nenhum ciclo em suas conexões, isto é, a informação sempre se move na mesma direção, da Input Layer para Hidden Layer(s), e daí, para Output Layer. Os pesos de suas conexões são atualizados de acordo com cada uma das entradas. A soma do produto dos pesos e dos inputs são calculados nó-a-nó.

Já o método de Backpropagation é um método diferente de atualização de pesos na rede, pois os erros para atualização de pesos são calculados apenas no final(ou seja, ao atingir a output layer, ao invés de ser calculado nó-a-nó), e só após isso que atualiza-se os valores ”de baixo para cima” nas conexões da rede neural.

## 3. O Problema

Considerando as seguintes relações familiares (fatos) como conhecimento prévio BK,

| | | |
|---|---|---|
| 1. parent(pam,bob). | 2. parent(tom,bob). | 3. parent(tom,liz). |
| 4. parent(bob,ann). | 5. parent(bob,pat). | 6. parent(pat,jim). |
| 7. parent(ann,eve). | 8. male(tom). | 9. male(bob). |
| 10. male(jim). | 11. female(pam). | 12. female(liz). |
| 13. female(ann). | 14. female(pat). | 15. female(eve). |

Seja o conceito objetivo (target) a ser induzido hd(X) para significar que X tem uma filha. Temos conhecimento que o conjunto $e^+ = hd(bob), hd(ann)$ são exemplos positivos e que o conjunto $e^- = hd(pam), hd(pat)$ são exemplos negativos, sendo $E = \{e^+, e^-\}$. O agente inteligente Amao gera, para BK e $E$ as seguintes bottomclauses

candidatas a hipóteses por indução. Note que variáveis foram adicionadas em substituição a constantes. Por exemplo, o fato female(pam) se colocado no corpo de uma candidata a hipótese para a meta, gera a regra atômica $hd(pam) \leftarrow female(pam)$, cuja bottom clause será $hd(X) \leftarrow female(X)$ sendo $\{pam/X\}$ (X substitui pam).

$C01 - \ hd(X) \leftarrow male(X), parent(Y, X).$

$C02 - \ hd(X) \leftarrow male(X), parent(X, Y).$

$C03 - hd(X) \leftarrow female(X), parent(Y, X).$

$C04 - \ hd(X) \leftarrow male(X), parent(Y, X).$

$C05 - hd(X) \leftarrow female(X), parent(X, Y), female(Y).$

$C06 - \ hd(X) \leftarrow male(X), parent(Y, X), female(Y).$

$C07 - hd(X) \leftarrow male(X), parent(X, Y), female(Y).$

$C08 - hd(X) \leftarrow female(X), parent(Y, X), male(Y).$

$C09 - hd(X) \leftarrow female(X), parent(X, Y), female(Y).$

$C10 - \ hd(X) \leftarrow male(X), parent(Y, X), parent(Y, Z).$

$C11 - hd(X) \leftarrow male(X), parent(X, Y), female(Y).$

$C12 - hd(X) \leftarrow male(X), parent(X, Y), female(Y), parent(Y, Z), male(Z).$

$C13 - hd(X) \leftarrow female(X), parent(Y, X), male(Y), parent(Z, Y), female(Z).$

$C14 - hd(X) \leftarrow female(X), parent(Y, X), male(Y), parent(Y, Z), female(Z).$

$C15 - hd(X) \leftarrow female(X), parent(Y, X), male(Y), parent(Y, Z), female(Z), parent(Z, W), male(W).$

1. Quais cláusulas podem gerar os exemplos negativos?

2. Utilizando o método do CILP++ apresentado no artigo [1], transformar essas cláusulas em uma rede neural (usando a ferramenta WEKA).

3. Treinamento da rede para os exemplos positivos e negativos, com outros exemplos que não estão em $e^-$ mas poderiam ser considerados.

4. Semelhante à Figura 2 do artigo, realizar o desenho de rede própria e mostrar as conexõees ativas para exemplos positivos e para exemplos negativos.

## 4. Soluções

### 4.1. Validação de cláusulas

**C01** - Y é pai/mãe de X. X é homem. Não se sabe se X tem uma filha, portanto, é indeterminado e pode gerar exemplos negativos. hd(X) == **false**.

**C02** - X é pai de Y. Não se sabe se Y é mulher ou homem. Logo, é indeterminado e pode gerar exemplos negativos. hd(X) == **false**.

**C03** - Y é pai/mãe de X. Mas X é Mulher. Logo, X tem filha. hd(X) == **true**.

**C04** - Y é pai/mãe de X. Mas X é Homem. Logo, Y tem um filho, mas não se sabe se Y tem uma filha, o que pode gerar exemplos falsos. hd(X) == **false**.

**C05** - X é mulher. X é mãe de Y. Mas Y é mulher. Logo, Y é filha de X. Portanto, X tem filha. hd(X) == **true**.

**C06** - X é homem. Y é pai/mãe de X. Mas Y é mulher. Logo, Y é mãe de X. Não está definido se Y tem filha, o que pode gerar exemplos falsos. Portanto, hd(X) == **false**.

**C07** - X é homem. X é pai de Y. Y é mulher. Logo, X tem uma filha. hd(X) == **true**.

**C08** - X é mulher. Y é pai/mãe de X. Y é homem. Mas sem necessidade da segunda cláusula, já sabiamos que Y tem filha. hd(X) == **true**.

**C09** - X é mulher. X é pai/mãe de Y. Mas Y é mulher. Logo, X tem filha. hd(X) == **true**.

**C10** - X é Homem. Y é pai/mãe de X. Mas também, Y é pai/mãe de Z. Mas não se sabe o sexo de Z. Portanto, pode-se gerar exemplos negativos. hd(X) == **false**.

**C11** - X é homem. X é pai de Y. Mas Y é mulher. Ou seja, X tem filha. hd(X) == **true**.

**C12** - X é homem. X é pai de Y. Y é mulher. Aqui, já sabemos que X tem filha, hd(X) == true. As demais definições são inúteis à pergunta hd(X), onde hd(X) == **true**.

**C13** - X é mulher. Y é pai/mãe de X. Y é homem. Z é pai/mãe de Y. Z é mulher. Mas em nenhum momento é dito que X é mãe. Logo, hd(X) == **false** pois não está definido se X tem filha.

**C14** - X é mulher. Y é pai/mãe de X. Y é homem. Y é pai de Z. Z é mulher. Mas em nenhum momento é dito que X é mãe. Logo, hd (X) == **false** pois não está definido se X tem filha.

**C15** - Dentre as clásulas apresentadas, em nehuma é dito que X é mãe. Logo, hd(X) == **false**.

### 4.2. Desenvolvimento da Rede Neural

De modo a tornar possível o treinamento das cláusulas pela rede neural, precisamos, inicialmente converter as cláusulas em tabelas representativas chamadas de datasets. Em seguida levando em consideração algumas regras estipuladas pelo algortimo CILP++, definimos a estrutura da rede em si.

#### 4.2.1. Modelagem do dataset

Durante a modelagem, cada cláusula foi transformada em um vetor numérico que pode ser processado por um algoritmo de aprendizado usando redes neurais. Isto foi implementado como segue:

1. Sendo N um número de literais distintos no corpo de uma cláusula E de uma lista de cláusulas L;
2. Um vetor de zeros é criado para representar a cláusula E, em que a quantidade de posições dos zeros representam todos os possíveis literais e a cabeça da cláusula;
3. Para cada literal presente em E, o zero é substituído por 1, de tal forma que a soma de todas as posições do vetor no corpo é N;

4. Para a posição do vetor que representa a cabeça de E, 0 é substituído por 1 apenas para o caso em que a proposição é considerada verdadeira;
5. O mesmo processo é repetido para cada cláusula de L;

aplicação deste passo a passo resultou no DataSet utilizado pela equipe, tabelado em uma planilha[4]. Lembrando que este DataSet foi utilizado para treinar a rede neural.

### 4.2.2. Definição dos hiperparâmetros da rede

Alguns hiperparâmetros (características fundamentais da rede) foram designados de acordo com orientações do algortimo CILP++, outros não especificados foram configurados com bases nas melhores práticas de.

1. Na camada de entrada há um neuron para cada possível literal do dataset, também pode ser dito que cada literal representa uma feature a ser treinada pela rede;
2. A camada escondida possui um neuron para cada cláusula do dataset;
3. Segundo instruções do CILP++, a camada de saída deve ter um neuron para cada literal existente na cabeça das cláusulas do dataset, e como queremos representar apenas $hd(X)$ ('has daughter') isso significaria apenas um neuron de saída. Porém, como a resposta é binária (verdadeiro ou falso) e precisamos representar tanto a probabilidade de uma dada entrada ser verdadeira como de que seja falsa, precisamos de no mínimo dois neurons na camada de saída.
4. Utilizamos uma taxa de aprendizado de 0.05 e o algortimo de otimização Adam Optimization, uma extensão do gradiente descendente estocástico que mantém uma taxa de aprendizado para cada peso da rede e os atualiza separadamente à medida que aprendizado se desenrola;
5. Para a função de ativação de cada neuron, utilizamos a função retificadora, ou "Unidade Retificada Linear(Rectified Linear Unit, ou ReLU)", uma função de rampa análoga à função retificadora de meia onda da engenharia elétrica, sua escolha se deve à demonstração em 2011[6] de que a utilização desta resulta em um melhor treino se comparada à função sigmoidal ou à tangente hiperbólica, uma aproximação da função é como segue:

$$f(x) = log(1 + e^x)$$

## 4.3. Treinamento da rede neural

O treinamento e avaliação da rede neural se deu através dos seguintes passos:

1. O número de cláusulas descrito no enunciado do projeto não se fazia suficiente para o treinamento confiável da rede, então criamos mais exemplos até termos por volta de quarenta exemplos (vinte verdadeiros e vinte falsos, divididos em dois datasets).
2. Os datasets com as cláusulas foram importados em formato .csv para o Tensorflow e os dados foram concatenados, misturados e separados em dados de entrada e etiquetas(labels).
3. Uma porcentagem de noventa por cento dos dados foi alocada para treinamento (35 cláusulas) e os dez por cento restantes (5 cláusulas) foram alocadas para o teste com a rede treinada.

4. As labels foram convertidas em vetores one-hot, que exibem 1 na posição relacionada à classe desejada (classe verdadeiro e classe falso) e 0 em todas as outras classes para simplificar o treinamento.
5. As cláusulas foram então treinadas pela rede por 15 épocas, e seu resultado foi comparado com a saída esperada para inferir a eficiência da rede, ao que obtivemos cem por cento de eficiência no treino e oitenta por cento no teste (4 exemplos certos e 1 errado).

## 4.4. Modelo da rede neural

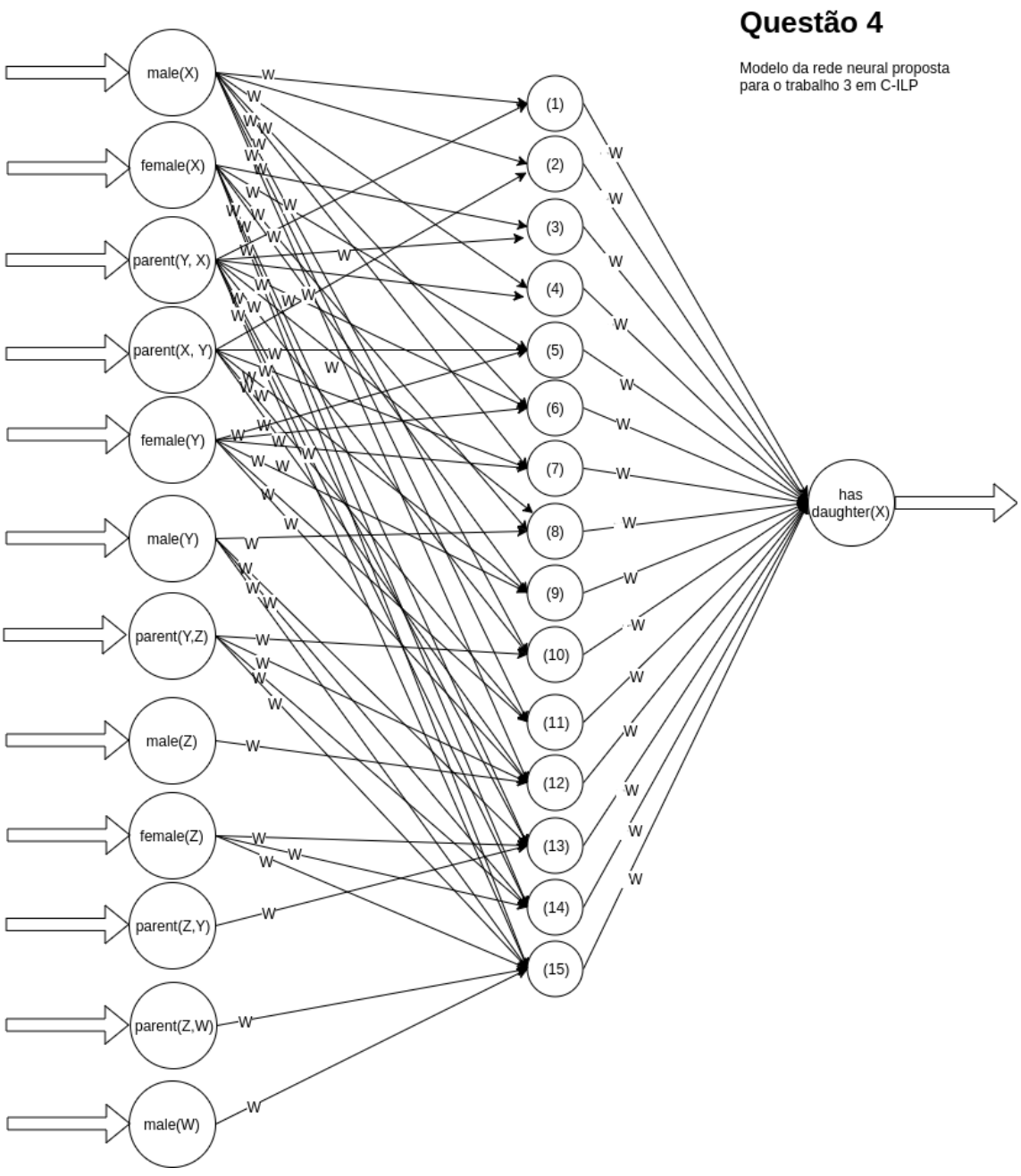


**Figure 4. Modelo da rede neural implementada**

## 5. Experimentos

Analisando os resultados obtidos pela rede, podemos considerar que temos um bom classificador. Os resultados são exibidos como se segue:

```
Entrada:  [[0 1 1 0 0 1 1 0 1 0 1 0]]
Saída da rede:  [[ 0.60760462 -0.90563357]]  - Label:  [[1 0]]
Predicao:  hd(X) falso  - Label:  hd(X) falso

Entrada:  [[1 0 0 1 0 1 1 1 0 0 0 0]]
Saída da rede:  [[-1.2520473   0.94565725]]  - Label:  [[1 0]]
Predicao:  hd(X) verdadeiro  - Label:  hd(X) falso

Entrada:  [[0 1 1 0 0 1 1 0 1 0 1 1]]
Saída da rede:  [[ 0.69922864 -0.80386519]]  - Label:  [[1 0]]
Predicao:  hd(X) falso  - Label:  hd(X) falso

Entrada:  [[0 1 1 0 0 1 1 0 0 0 0 0]]
Saída da rede:  [[ 0.54710376 -0.54683727]]  - Label:  [[1 0]]
Predicao:  hd(X) falso  - Label:  hd(X) falso

Entrada:  [[1 0 0 1 1 0 1 1 0 0 1 1]]
Saída da rede:  [[-1.36420608  1.07734716]]  - Label:  [[0 1]]
Predicao:  hd(X) verdadeiro  - Label:  hd(X) verdadeiro
```

Os números exibidos ao lado de ”Saída da rede” referem-se aos outputs dos dois neurons de saída que indicam a probabilidade de uma cláusula com o corpo indicado - exibida aqui ao lado de ”Entrada” como um vetor de zeros e uns que representa os literais do corpo da cláusula - indicar verdadeiro (0,1) ou falso (1,0) na cabeça da cláusula. Estes valores são comparados com os resultados esperados para as cláusulas em questão, aqui representados ao lado de ”Label”. Logo abaixo exibimos os resultados de forma mais didática, em que ”Predição” significa o resultado atingido pela rede para hd(X) em comparação com ”Label”, ou o resultado esperado para a cláusula em questão.

## 6. Conclusões

Neste trabalho buscamos implementar e testar o método proposto por França[1], porém, devido à dificuldades de implementação do CILP++, recorremos à uma modificação utilizando Feedfoward ao invés de Backpropagation, ou seja, um método alternativo ao CILP++. Mesmo com tal particularidade, fomos bem sucedidos em nossos testes, adquirindo resultados esperados em relação à proposta inicial(a de implementação do CILP++). Fora utilizado um dataset com 40 exemplos, sendo 35 para treinamento da rede neural e 5 para teste[4].No futuro, poderia ser feito um novo treinamento com um dataset maior, reduzindo assim o overfitting e melhorando o desempenho do sistema como um todo. Com isso, consideramos que modelos de construção e implementação alternativos também produzem resultados aceitáveis ao longo do tempo, desconsiderando, por hora, possíveis questões de desempenho entre os dois métodos.

## 7. Referências

[1] França, M.V.M., Zaverucha, G., d'Avila A Garcez, A.S.: Fast relational learning using bottom clause propositionalization with artificial neural networks. Machine Learning 94(1), 81–104 (Jan 2014)

[2] Garcez, A. S. D., & Zaverucha, G. (1999). The connectionist inductive learning and logic programming system. Applied Intelligence, 11, 59–77.

[3] RUSSEL, S.; NORVIG, P.; Inteligência Artificial. Elsevier, Terceira Edição, 2013.

[4] Hauache N., Bonfim A., Neto E.: Base de Dados. Disponível em: https:docs.google.comspreadsheetsd1Re2OhAMJDBbp0OByq-EUyKZdDdEZEaFVy_iNJgbVl9Uedit?ts=5b3d3410#gid=0.

[5] Mitchell, Thomas M. (1997). Machine Learning.

[6] Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011). Deep sparse rectifier neural networks . AISTATS.

[7] Mota,Edjard de S.;Howe, Jacob M.;Garcez, Artur S. D'Ávila. Inductive Learning in Shared Neural Multi-Spaces.