ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Thearlismar Soares de Araújo

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA O DIAGNÓSTICO DE PATOLOGIAS TRAUMATO-ORTOPÉDICAS DOS MEMBROS INFERIORES

Palmas

2012

Thearlismar Soares de Araújo

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA O DIAGNÓSTICO DE PATOLOGIAS TRAUMATO-ORTOPÉDICAS DOS MEMBROS INFERIORES

Trabalho apresentado como requisito parcial da disciplina de Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pelo Professor Mestre Fabiano Fagundes.

Palmas - TO

2012

Thearlismar Soares de Araújo

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA O DIAGNÓSTICO DE PATOLOGIAS TRAUMATO-ORTOPÉDICAS DOS MEMBROS INFERIORES

Trabalho apresentado como requisito parcial da disciplina de Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pelo Professor Mestre Fabiano Fagundes.

Aprovada em xxxxxxx de 2012.

BANCA EXAMINADORA

Prof. M.Sc. Fabiano Fagundes

Centro Universitário Luterano de Palmas

Prof. M.Sc. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas

Prof. M.Sc. Cristina D'Ornellas Filipakis

Centro Universitário Luterano de Palmas

Palmas - TO

2012

SUMÁRIO

RESUMO

O diagnóstico de patologias, em específico as traumato-ortopédicas dos membros inferiores, é baseado em exames e testes para determinar sua origem. Mas existem vários exames e testes para indicar estas doenças, sendo assim a aplicação da técnica de Redes Neurais Artificiais (RNAs) é utilizada neste trabalho como forma auxiliar especialistas da área da fisioterapia traumato-ortopédica no diagnóstico destas patologias. A RNA é uma técnica que simula o processamento do cérebro humano, tendo a capacidade de aprender através de exemplos, é considerada uma técnica da Inteligência Artificial (IA). A IA é uma área capaz de resolver vários problemas, criando sistemas inteligentes, baseados no comportamento humano. O objetivo deste trabalho é desenvolver um sistema de diagnóstico de patologias traumato-ortopédicas dos membros inferiores aplicando a técnica de Redes Neurais Artificiais, criando e treinando a rede neural, além de validar a solução através de testes, com o auxílio e supervisão de um especialista do domínio.

PALAVRAS-CHAVE: Patologias ortopédicas, Redes Neurais Artificiais, Inteligência Artificial.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

AC – Aquisição de Conhecimento

BC – Base do Conhecimento

IA – Inteligência Artificial

SE – Sistema Especialista

RNA – Rede Neural Artificial

RNB – Rede Neural Biológica

ULBRA – Universidade Luterana do Brasil

JavaNNS – Java Neural Network Simulator

1 INTRODUÇÃO

Os membros inferiores são a base de sustentação do corpo humano, permitindo a locomoção e a postura. Sofrem impactos e pressões todos os dias, sendo expostos a desgastes e lesões. A área da fisioterapia estuda vários problemas relacionados a estes membros, buscando corrigi-los e solucioná-los. Os diagnósticos, nestes casos, são realizados através dos resultados apresentados nos exames ou testes e, de acordo com estes indicadores, chega-se a identificação do problema. Muitas vezes estes sintomas são parecidos em diferentes doenças, o que dificulta a identificação da patologia e faz com que especialistas busquem soluções para aprimorar seu diagnóstico.

A partir disso, este trabalho tem como objetivo desenvolver um sistema de diagnóstico de patologias traumato-ortopédicas dos membros inferiores aplicando a técnica de Redes Neurais Artificiais. Serão abordados conceitos como Inteligência Artificial (IA), Redes Neurais (RN) e suas características, como também as Patologias Traumato-Ortopédicas encontradas nos Membros Inferiores na área da Fisioterapia.

A IA é um campo que tem o objetivo de utilizar técnicas que representam o comportamento humano. A técnica de Redes Neurais Artificiais busca simular a capacidade de raciocínio, através da simulação das redes neurais do cérebro. Assim, utilizando sinais e sintomas apresentados por determinadas patologias e aplicando as RNAs buscar-se-á alcançar aproximação das patologias através do treinamento destas redes neurais.

Com isto, para auxiliar no desenvolvimento deste trabalho, em específico na criação da RNA e seu treinamento, foi realizada uma busca por ferramentas, sendo a *Java Neural Network Simulator* (JavaNNS) escolhida por já ser utilizada em trabalhos anteriores, similares a este, e também por possuir uma documentação detalhada. Com o auxílio desta ferramenta, foi criada e treinada a rede neural artificial para o problema em específico, permitindo relacionar sinais e sintomas com suas respectivas patologias. Assim, foi possível conseguir determinadas aproximações do resultado ideal através de conjuntos de entradas aleatórios e analisar os resultados obtidos pela rede neural artificial junto ao especialista do domínio. Na próxima seção serão apresentados os conceitos para compreensão do trabalho, destacando a Redes Neurais Artificial, seus conceitos e características

2 REFERENCIAL TEÓRICO

Nesta seção são apresentados os conceitos necessários para compreensão do trabalho. Abordando os conceitos de Inteligência Artificial (IA), Redes Neurais Artificiais (RNAs), *Java Neural Network Simulator* (JavaNNS), Fisioterapia, Fisioterapia Traumato-Ortopédica e as Patologias dos membros Inferiores.

2.1. Inteligência Artificial

Historicamente, a Inteligência Artificial surgiu após a Segunda Guerra Mundial, sendo considerada uma ciência recente e com um campo muito vasto para pesquisas que objetivam suprir as necessidades de compreensão do pensamento (RUSSELL, 2004, p. 3).

Os sistemas tradicionais que são os mais utilizados no cotidiano de milhares de pessoas, se diferenciam dos sistemas inteligentes. Segundo Rezende (2005, p. 4-7) sistemas ou aplicações tradicionais utilizam abordagens manuais para resolver tarefas, sendo alguns exemplos os processadores de texto e programas gráficos. Já os sistemas inteligentes têm a capacidade de trabalhar de modo associativo e não-linear, capazes também de trabalhar com grande quantidade de informação de maneira eficiente, utilizando das associações e inferência, similar ao processo de decisão humana.

A IA abrange uma variedade de subcampos, sistematizando e automatizando tarefas intelectuais, tornando-se um campo universal (RUSSELL, 2004, p. 3).

De acordo com Coppin (2010, p. 4) IA é definida pelas propriedades apresentadas por ela, como:

- capacidade de lidar com novas situações;
- solucionar problemas;
- responder a questões; e
- formar planos, a partir de previsões da rede neural.

Segundo Rezende (2005, p. 8), a IA possui várias técnicas e métodos utilizados para a criação de sistemas inteligentes que são muito importantes para soluções de problemas que exigem a tomada de decisões:

- Algoritmos Genéticos;
- Redes Neurais Artificiais;
- Lógica*Fuzzy*;
- Mineração de Dados e Texto;
- Sistemas Especialistas;
- Sistemas Baseados em Casos;
- Redes bayesianas;

Este trabalho irá se aprofundar na técnica de Redes Neurais Artificiais, pois em paralelo a este trabalho existem outros que se utilizam do mesmo domínio, mas com técnicas diferentes, e por isso a utilização desta técnica neste trabalho pode ser mais interessante, pois dependendo dos resultados adquiridos, pode se comparar aos outros trabalhos e definir qual técnica obteve melhor resultado. Na próxima seção será apresentado como são formadas as redes neurais biológicas que compõem o cérebro humano, para uma analogia com a técnica de redes neurais artificiais (RNAs).

2.2 Redes Neurais Biológicas

A Rede Neural Biológica (RNB) é constituída por milhares de neurônios interligados entre si e são a base do sistema nervoso, já que possuem a propriedade de excitação e condução de impulsos nervosos. O sistema nervoso é altamente complexo, porém somente através destas estruturas que é possível ter a capacidade de controlar e coordenar os processos vitais, de forma voluntária ou involuntária, através das vias sensitivas e motoras. Estas vias são o meio do sistema nervoso captar informações do ambiente através de sensores como a pele e responder por reações motoras como fechar os olhos como exemplo (ARONE, 2005, p.11).

Quando esses neurônios se conectam e criam a Rede Neural, é possível processar uma grande quantidade de informação. “Um neurônio é uma célula no cérebro cuja principal função é coletar, processar e disseminar sinais elétricos” (RUSSELL 2004, p. 713). Coppin (2010, p. 254) descreve que os neurônios são elementos de processamento muito simples, e o cérebro humano possui cerca de mais de 10 bilhões de neurônios, conectados a milhares de outros formando cerca de 60 trilhões de conexões, que é a base de todo o potencial do ser humano.

De acordo com Braga (2010, p. 5), em uma Rede Neural Biológica (RNB) cada neurônio é dividido em três principais seções: o corpo da célula, os dendritos e o axônio, como pode ser observado na Figura 1.

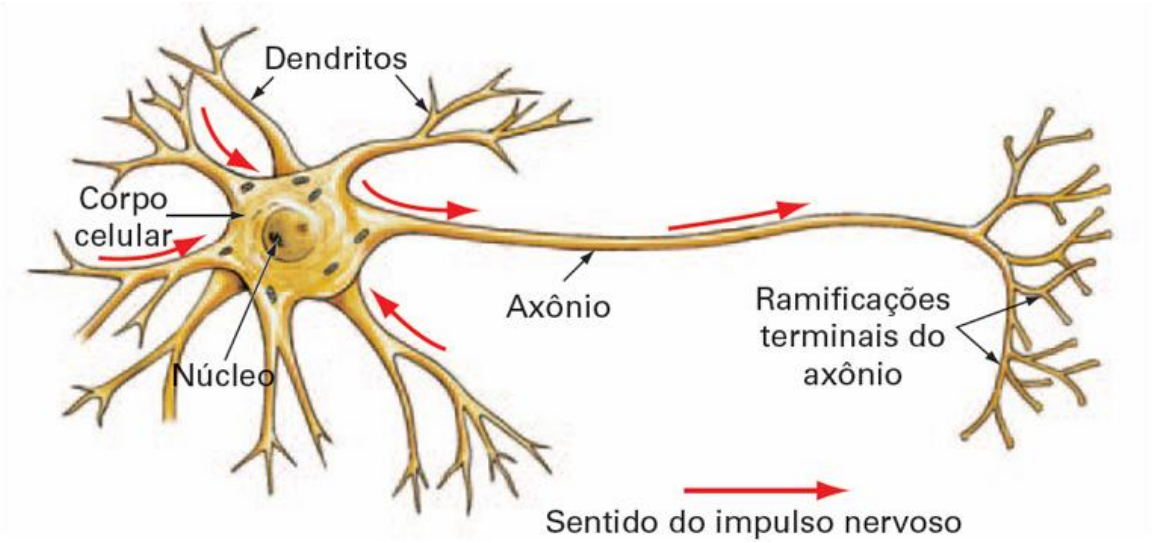

Figura 1 - Representação do neurônio biológico. (ENTENDA..., 2012, p. 1)

Na Figura 1 pode ser observada a representação do neurônio biológico, onde os impulsos nervosos são receptados através dos dendritos que percorrem o núcleo envolvido pelo corpo celular, passando pelo axônio até seus terminais para formarem as sinapses, que formam as interações entre os neurônios (HAYKIN, 2001, p. 32-33).

Segundo Arone (2005, p.12) "as sinapses são os pontos de conexão entre os neurônios através das ligações que ocorrem entre as ramificações dendríticas, tendo como fio condutor o axônio". As sinapses basicamente são as ligações entre os neurônios, porém não existe necessariamente um contato entre estes neurônios, mas estímulos ou impulso nervoso que percorre os neurônios porque uma série de modificações químicas e elétricas ocorre no local de comunicação entre eles. A figura 2 ilustra como ocorrem as sinapses entre os neurônios.

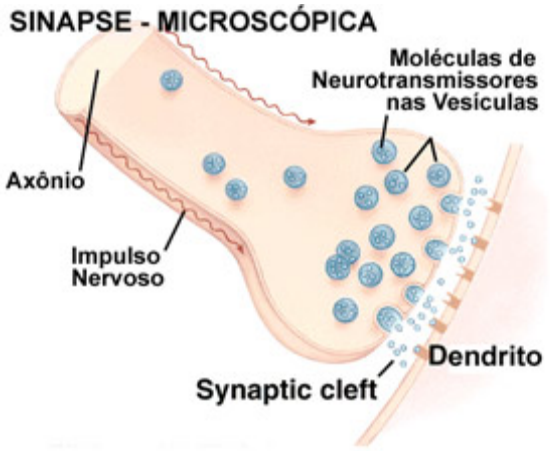

Figura 2 – Ilustração do processo da sinapse. (BALLONE, 2012, p. 1)

Na figura 2 é observado o processo da sinapse entre os neurônios, não existem ligações físicas entre ambos, mas impulsos dos terminais do axônio de um neurônio com os dendritos de outro. Na rede neural existem milhares de sinapses ocorrendo entre neurônios de modo paralelo, que proporciona a capacidade de racionalização dos seres vivos.

A Inteligência Artificial inspirada neste funcionamento possui uma técnica capaz de representar a rede neural biológica, denominada Redes Neurais Artificiais, apresentada na próxima seção.

2.2. Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são inspiradas no funcionamento do cérebro humano, ou seja, na rede neural biológica. Ela é definida como sendo uma forma de computação não-algorítmica, por não ser baseada em regras ou programas, sendo uma alternativa à computação algorítmica convencional (BRAGA, 2000, p.1).

As RNAs são similares as Redes Neurais Biológicas (RNBs), pois a base de seu funcionamento é o neurônio. Segundo Haykin (2001, p. 36) o neurônio é uma unidade de processamento de informação fundamental para a operação de uma rede neural. Entretanto, de acordo com Coppin (2010, p. 255) as RNAs são menores e possuem neurônios artificiais com menos conexões em termos quantitativos relacionados às redes neurais e neurônios biológicos. Bocanegra (2002, p. 7) expõe também que o Neurônio Artificial é a unidade fundamental de processamento de uma RNA, o qual recebe uma ou mais entradas, transformando-as em saídas.

A figura 3 ilustra o modelo de neurônio artificial proposto por McCulloch e Pitts em 1943.

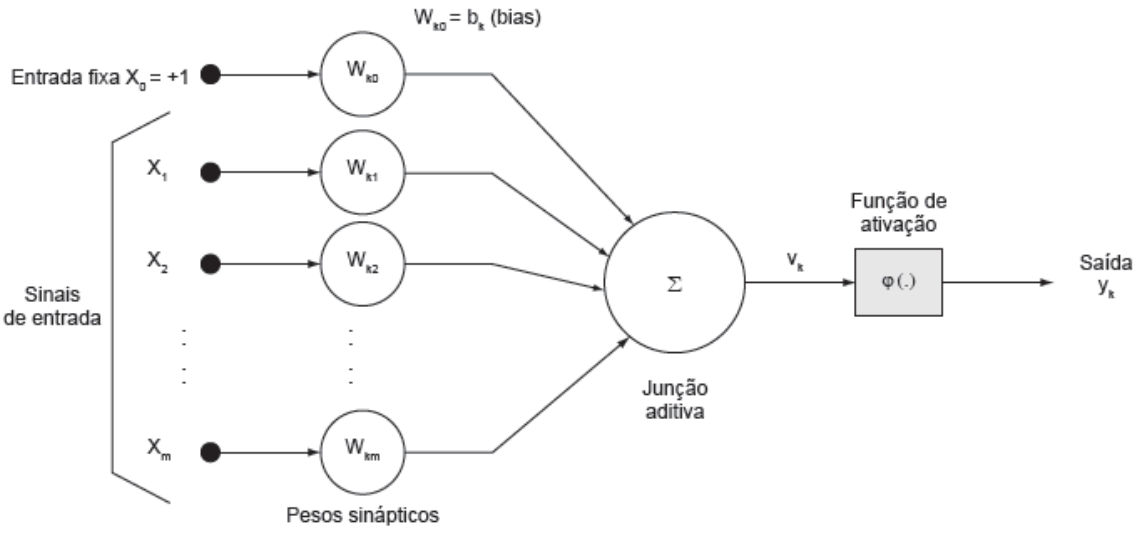

Figura 3 - Funcionamento do neurônio artificial, baseado em (HAYKIN, 1999, p. 38).

Na figura 3 é observado o funcionamento de um neurônio artificial, onde X são os estímulos de entrada, e W as ligações sinápticas. As ligações recebem um peso representando seu potencial, chamado de pesos sinápticos, onde cada entrada vai ser multiplicada pelo peso de sua ligação, gerando um sinal tanto positivo ou negativo, sendo um sinal positivo considerado excitatório e, no caso contrário, inibitório.

De acordo com Neto (2006, p. 62), as RNAs possuem quatro etapas de implementação que constituem seu ciclo de vida: definição da rede, treinamento, utilização da RNA e manutenção, como pode ser observado na figura 4.

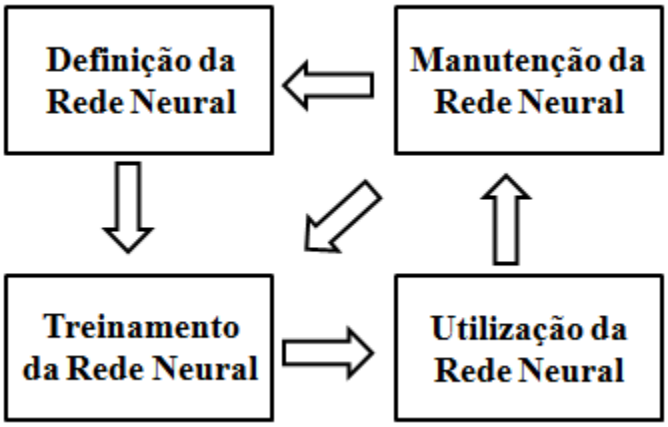

Figura 4 - Ciclo de Vida de uma RNA, baseado em (NETO, 2006, p.62).

Na figura 4 pode ser visto o ciclo de vida de uma RNA, em que pode ser observada que a primeira etapa é a definição da rede neural. Nesta fase são definidos o propósito da rede, o tipo de topologia que será utilizado, o tipo de treinamento e os dados de entrada. Logo se passa para a etapa de treinamento da rede, onde são inseridos os dados de entrada e feito o treinamento de acordo com as definições da etapa anterior.

Posteriormente vem a etapa de utilização da rede onde são extraídos os dados de saída, que são verificados e comparados; se a rede não atingir um valor próximo a seu objetivo segue para a próxima fase. Já na etapa de manutenção da rede, são feitos os ajustes necessários para que a rede seja treinada novamente e forneça dados mais confiáveis, caso exista a necessidade de alteração de seu modelo ou proposta, o que inicia outra vez a etapa de definição da rede, retomando o ciclo novamente.

Um dos principais objetivos das redes neurais são modelos com boa capacidade de generalização, baseando no conjunto de dados utilizados na rede. Tendo determinado valor para a entrada e para uma saída desejável, a rede deve ser ajustada e treinada para que a diferença entre a saída da rede e a saída ideal, que é o erro, diminua a cada interação para se conseguir uma melhor generalização da rede (KARRER, 2005, p. 6 *apud* BRAGA, 2000, p.1). Na próxima seção serão apresentadas as funções de ativação das redes neurais.

2.3.1 Funções de Ativação

A função de ativação define a saída do neurônio artificial e é aplicada nas entradas de uma rede neural, resultando no nível de ativação do neurônio, o que determina se um neurônio irá ativar ou não. Se este neurônio ativar, será utilizado para definir a saída da rede, caso contrário, estará inativo, e não interfere na saída da rede. A função de ativação basicamente controla o estado do neurônio. Braga (2000, p. 10) apresenta alguns gráficos (apresentados nas próximas seções) para as funções de ativação e equações que as representam e que, quando utilizadas, resultam em um faixa de saída para o neurônio.

2.3.1.1 Função Linear

Segundo Coppin (2010, p. 256) a função linear é uma das mais frequentes encontradas em RNAs e, basicamente, é a soma ponderada das entradas da rede com o nível de ativação. É muitas vezes utilizada na primeira camada da rede e trabalha em conjunto com outras funções como a sigmoidal, pois apresenta uma saída com valores reais. Esta função pode ser

observada pela figura 5.

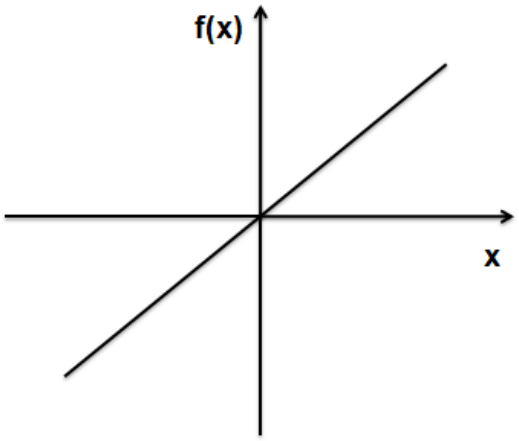

Figura 5 - Função de ativação linear. Baseado em (BRAGA, 2000, p. 10).

A figura 5 é definida pela equação abaixo, apresentando o resultado e a saída da rede neural.

$$y = \alpha x$$

Na equação o valor de y é a saída da rede neural, que é o produto entre x que é o valor de entrada e α, o nível de ativação, que é uma constante que representa a saída linear para os valores de entrada, segundo Oliveira (2007, p. 19).

2.3.1.2 Função Rampa

Segundo Braga (2000, p. 10) a função Rampa possui características que restringem seu sinal a valores constantes em uma determinada faixa, como 1 ou 0, sem meios termos (Figura 6).

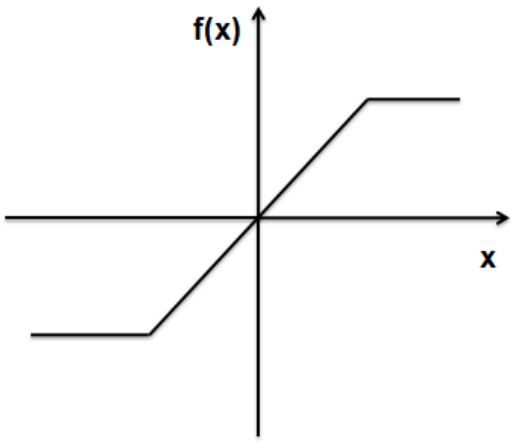

Figura 6 - Função de ativação rampa. Baseado em (BRAGA, 2000, p. 10).

A figura 6 é definida pela equação abaixo, onde é determinada a faixa de valores da função, como o máximo $+\gamma$ e o mínimo $-\gamma$ para determinar a saída da rede neural. Essa limitação aplicada em uma função linear é um meio para se obter valores constantes e transformar a função linear em função de rampa.

$$y = \begin{cases} +\gamma \; se \; x \geq +\gamma \\ x \; se \; |x| < +\gamma \\ -\gamma \; se \; x \leq -\gamma \end{cases}$$

Na equação são apresentadas as condições de saída da função. Caso x seja maior ou igual a $+\gamma$ este será a saída. Em outro caso, se o módulo de x for menor que $+\gamma$ a saída será x. E a última condição: se x for menor ou igual a $-\gamma$, esta será a saída.

2.3.1.3 Função Passo

A função Passo é conhecida também como função de degrau, pela característica de suas formas, e é utilizada no modelo de McCulloch e Pitts observado na figura 2. Basicamente produz uma saída binária, um número real, como pode ser observado na figura 7.

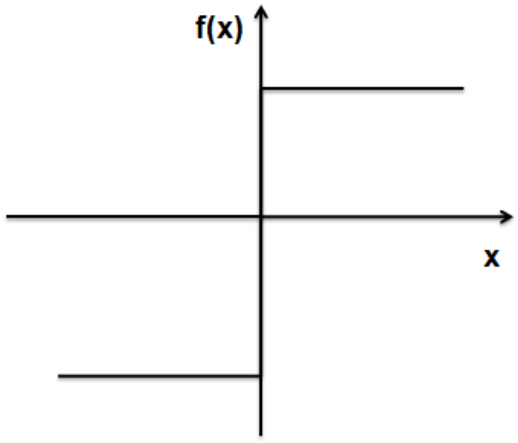

Figura 7 - Função de ativação passo. Baseado em (BRAGA, 2000, p. 10).

A figura 7 é definida pela equação abaixo, onde são apresentados o resultado e a saída da rede neural.

$$y = \begin{cases} +\gamma \; se \; x > 0 \\ \\ -\gamma \; se \; x \leq 0 \end{cases}$$

Na equação é feito a comparação do valor de x para determinar a saída. No caso, para a situação em que x seja maior que 0 (zero), a saída será $+\gamma$, caso contrário e se for igual à saída será $-\gamma$ definido nos parâmetros da equação, nesta função seu valor de saída apresenta 1 em caso de ativação ou -1 para a inibição, por isso a característica de passo, pois ou vai pra frente ou pra trás, no caso análogo do neurônio ou ativa ou permanece em repouso que é caso do sinal inibitório.

2.3.1.4 Função Sigmoidal (Sigmóide)

Segundo Haykin (1999, p. 40) a função de ativação sigmóide é muito comum na construção de redes neurais artificiais e é utilizada por vários modelos de aplicações. Braga (2000, p. 11) define como sendo uma função semilinear, limitada e monotônica e que também é conhecida como S-*shape*. A função de ativação pode ser observada na figura 8.

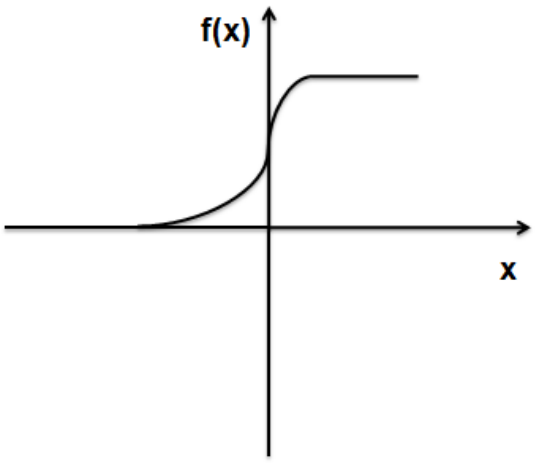

Figura 8 - Função de ativação sigmoidal. Baseado em (BRAGA, 2000, p. 10).

A figura 8 é definida pela equação abaixo. Na equação y é a saída a que podem ser atribuídos valores que variam de 0 a +1.

$$y = \frac{1}{1 + \varepsilon^{-x/T}}$$

O que se destaca na equação é que, segundo Braga (2000, p. 11), o parâmetro que determina a suavidade da curva da função sigmoidal é T, o que, segundo Oliveira (2007, p. 21) representa o ruído (interferência) sináptico. Esta função resulta em valor entre 0 e 1.

2.3.2 Arquitetura das Redes Neurais Artificiais

Existem várias arquiteturas de RNAs, mas estas basicamente podem ser divididas quanto ao número de camadas, e pelos tipos de conexões entre os neurônios. A arquitetura de RNAs, definida pelo número de camadas, pode ser RNAs de camada única ou múltiplas camadas.

Segundo Mathias (2006, p. 19) as RNAs com camada única são a forma mais simples de uma rede em camadas e surgem quando se tem uma camada de entrada que se projeta para a camada de saída, como mostrado na figura 9.

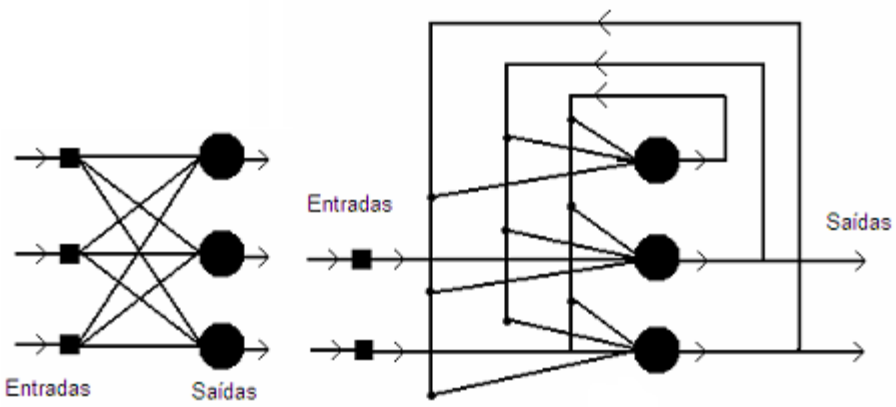

Figura 9 - Redes de camada única (MATHIAS, 2006, p. 19).

Na figura 9 é ilustrado o exemplo de redes de camada única. Estas redes apresentam uma estrutura simples de entradas e saídas, utilizadas em problemas de classificação, ordenação. As redes de camada única não apresentam camadas ocultas, as camadas ocultas aumentam as conexões entre os neurônios, fazendo a rede analisar mais variáveis até chegar a um resultado.

As RNAs com múltiplas camadas distinguem-se de redes com camada única pela presença de camadas ocultas. A função das camadas ocultas é extrair informações das amostras, conforme apresentado na figura 10 (MATHIAS, 2006, p. 19).

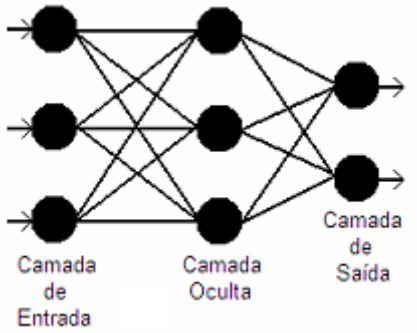

Figura 10 - Rede com múltiplas camadas (MATHIAS, 2006, p. 19).

Na figura 10 ilustra a rede com múltiplas camadas, utilizada em problemas mais complexos, que necessitam de analisar várias combinações das entradas da rede, como o diagnóstico de doenças, em que necessita-se verificar a relação dos sinais e sintomas antes de determinar qual saída mais provável pela rede.

A arquitetura das RNAs também pode ser definida pelos tipos de conexões entre neurônios, sendo denominadas como *Feedforward* e *Feedback*. *Feedforward*, acíclicas ou não-recorrentes são estruturadas em camadas, não possuindo realimentação em suas saídas para as entradas, que podem ter de uma ou mais camadas em sua configuração (BOCANEGRA, 2002, p. 12). Pode ser observado na figura 11 um exemplo de RNA não recorrente, em que os pares de entrada passam pela camada de rede até a camada de saída.

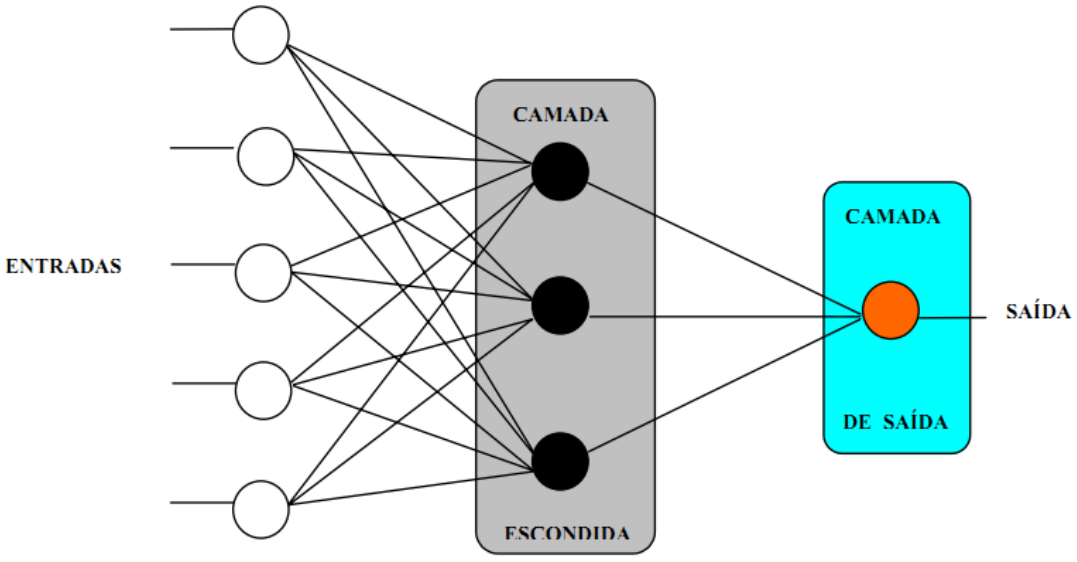

Figura 11 - Exemplo de uma RNA não recorrente (BOCANEGRA, 2002, p. 12).

A figura 11 apresenta o modelo de uma RNA não recorrente que, basicamente, é formada pelo conjunto de entradas da rede interligadas a camada escondida que, por sua

vez, é interligada à camada de saída, gerando a saída da rede. As entradas da rede serão combinadas, através das conexões, até a rede determinar a saída mais provável, para o conjunto de entradas. De fato não existe uma ligação da saída com as entradas, formando um ciclo, como ocorre nas RNAs *Feedback*.

As RNAs *Feedback*, cíclicas ou recorrentes, diferenciam-se das não-recorrentes por terem em sua estrutura a realimentação das saídas para as entradas. Outra característica é que sua estrutura não é organizada em camadas, ainda que existam casos em que tal configuração ocorre (BOCANEGRA, 2002, p. 13). A figura 12 apresenta o modelo da estrutura de RNA recorrentes.

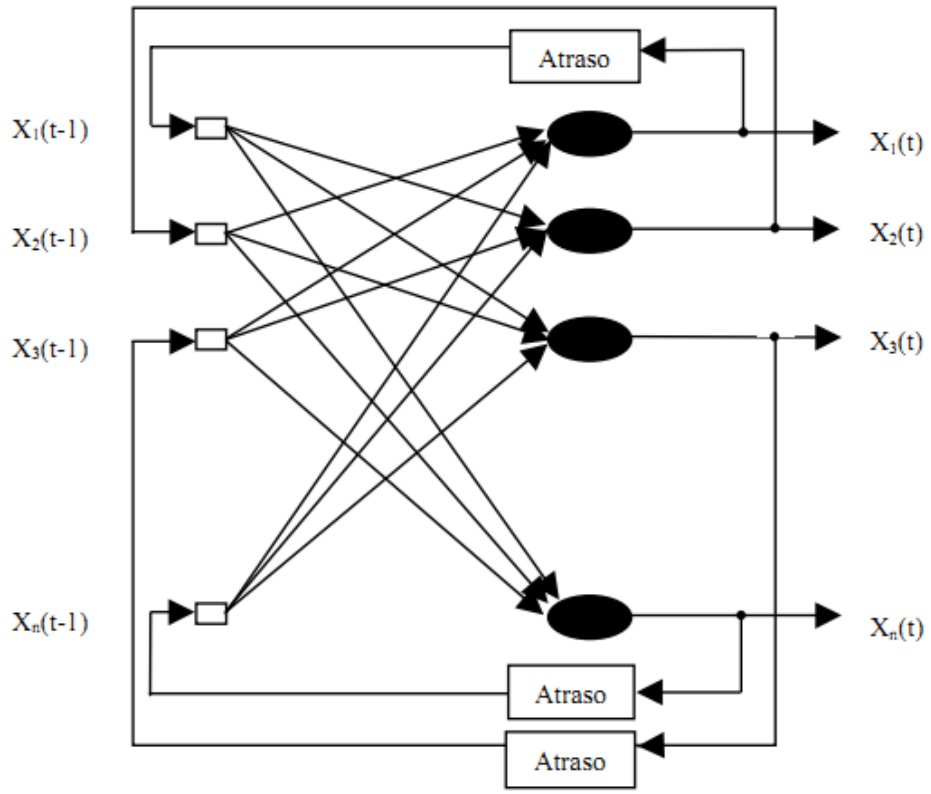

Figura 12 - Exemplo de uma RNA recorrente (BOCANEGRA, 2002, p.13).

No modelo apresentado pela figura 12, a RNA tem todas suas saídas ligadas ao conjunto de entradas da rede, numa tentativa de correção das entradas em eventuais erros. A rede utiliza suas saídas para buscar um resultado, em um processo de refinamento constante.

Neste trabalho a arquitetura utilizada nas RNAs foi a de múltiplas camadas acíclica (*Feedforward*). Este tipo de arquitetura foi escolhido por trabalhar com o aprendizado supervisionado, apresentado na próxima seção.

2.3.3 Aprendizado

Uma das características mais interessantes das RNAs e uma das mais importantes é a capacidade de aprender através de exemplos, o que faz evidenciar sua semelhança com o comportamento humano. Outra característica importante é a possibilidade de se adaptar a problemas específicos, fazendo interpolações e extrapolações do que aprendem.

As ligações são semelhantes às sinapses, agrupando características para determinar algo, como, por exemplo, quando se aprende a escrever, no começo é complicado, pois é necessária uma junção de letras para formar determinada palavra, porém, com a prática, o cérebro cria conexões que melhoram o desempenho..

Para que uma RNA possa solucionar determinadas tarefas, ela deve ser submetida à fase de aprendizagem. Nesta fase a RNA busca extrair as informações e padrões relevantes de determinados problemas, para que possa criar uma representação própria para o problema (BRAGA, 2000, p 15).

Segundo Rezende (2005, p. 158) a obtenção de modelos com capacidade de generalização de um conjunto de dados é um dos principais objetivos do aprendizado em RNA. A generalização é uma das melhores formas de cobertura dos dados. Para que uma RNA possa aprender, seus parâmetros ou configurações da rede devem passar por um processo de ajustes contínuos de estímulos do ambiente em que está operando (BRAGA, 2000, p. 15).

O algoritmo de aprendizado é o responsável por ajustar os parâmetros da RNA, onde a convergência da solução, ou seja, conseguir chegar ao resultado esperado, dá-se pelo conjunto determinado de iterações. O objetivo destes ajustes contínuos é que a RNA apresente uma mudança gradual em seu comportamento e desempenho (REZENDE, 2005, p. 149).

Ocorre também a diferença na maneira em que os algoritmos de aprendizado obtêm os ajustes necessários as RNA. Segundo Rezende (2005, p. 149),o aprendizado de RNAs pode ser classificado em três paradigmas distintos: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço, que serão apresentados nas próximas seções.

2.3.3.1 Aprendizado Supervisionado

O aprendizado supervisionado é como se existisse um aluno que é cobrado para chegar a um determinado resultado, mas que também tem um professor que o auxilia nesta tarefa,

corrigindo-o até alcançar o objetivo esperado. Este é o método mais comum em treinamentos de RNAs e caracteriza-se pela presença de um supervisor externo que fornece as entradas e saídas desejadas da rede. O objetivo deste treinamento é encontrar uma ligação entre os pares de entrada e saída fornecidos pelo supervisor, ajustando os parâmetros da rede (BRAGA, 2000, p. 16).

A desvantagem do aprendizado supervisionado é que a RNA não consegue aprender estratégias para novas situações que não estejam nos exemplos de treinamento da rede quando o supervisor está ausente (BRAGA, 2000, p. 16).

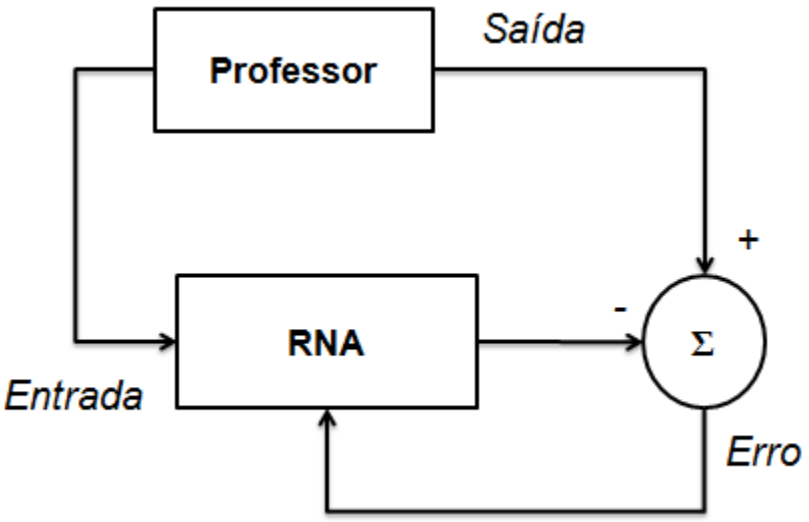

Figura 13 - Aprendizado supervisionado. Baseado em (BRAGA, 2000, p. 17).

A figura 13 apresenta o mecanismo de aprendizado supervisionado durante o processo de treinamento da RNA. O professor indica a entrada para a rede e supervisiona sua saída, analisando seu comportamento. Toda saída da rede é monitorada e, dependendo do resultado, são feitos os ajustes necessários para diminuição dos erros, que é a diferença da saída que se almeja alcançar. Todo esse processo ocorre várias vezes gerando ciclos até encontrar a saída desejada.

2.3.3.2 Aprendizado Não-supervisionado

O aprendizado não supervisionado é usado normalmente em problemas de categorização de dados, determinando que um conjunto de dados pertença a uma classe. É caracterizado por não haver a presença de um supervisor que fornece as saídas desejadas para as entradas

e acompanha o processo de aprendizado da rede. Assim, não há o supervisor que ajusta os parâmetros, sendo os ajustes obtidos somente através do conjunto de valores de entrada (REZENDE, 2005, p. 150).

A figura 14 apresenta o modelo do aprendizado não-supervisionado, em que não existe a presença de um supervisor analisando as saídas, e nem mesmo fornecendo as entradas da rede.

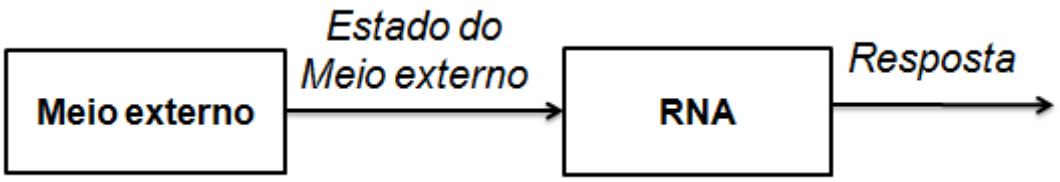

Figura 14 - Aprendizado não-supervisionado. Baseado em (BRAGA, 2000, p. 19).

Na figura 14, as entradas que a rede recebe são representadas pelo meio externo ou um conjunto qualquer de dados, fornecidos a RNA. Com essas entradas, a RNA estabelece critérios para estes dados, permitindo classificá-los, ou orientá-los para um conjunto de saídas possível. A rede neural busca analisar a similaridade das entradas para ter uma compreensão do que estão sendo fornecido. Em todo o processo não existe um monitoramento de um supervisor, o qual verificaria as saídas.

2.3.3.3 Aprendizado por Reforço

O aprendizado por reforço, mesmo apresentando similaridades com o aprendizado supervisionado e não supervisionado, é diferente, pois é caracterizado por haver um supervisor que não indica uma saída desejada, apenas avalia se a saída está correta ou não (REZENDE, 2005, p. 150).

Na figura 15 pode ser observado o modelo de aprendizado por reforço. Não possui um supervisor, mas um crítico que aplica penalidades ou reforço, dependendo do resultado apresentado pela rede.

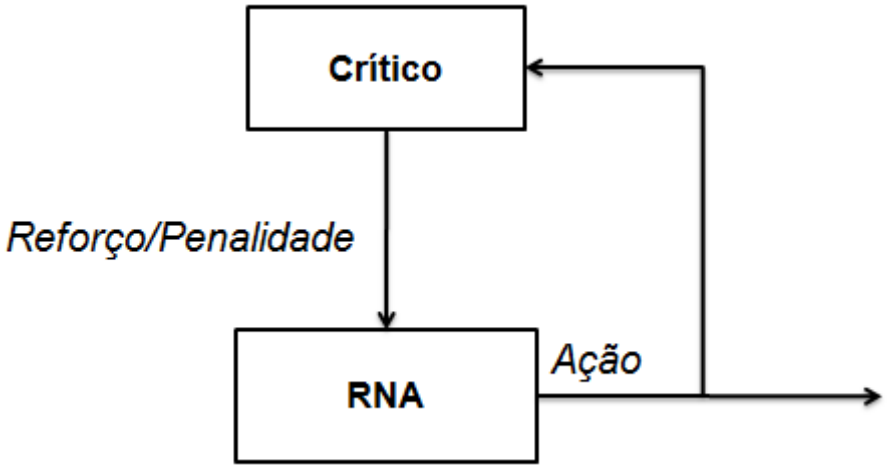

Figura 15 - Aprendizado por reforço. Baseado em (BRAGA, 2000, p. 26).

Na figura 15 nota-se que existem ciclos de operações que podem ser realizadas várias vezes até a rede apresentar uma saída mais próxima da esperada. Em sua estrutura as saídas da RNA são avaliadas por um supervisor que aplica um reforço ou penalidade a RNA. Este reforço é aplicado quando a rede apresenta uma saída mais próxima da esperada, o que faz com que o resultado ocorra com mais frequência na rede. Já a penalidade tenta inibir o resultado apresentado pela rede fazendo com que ocorra com menor frequência (REZENDE, 2005, p. 150).

Um exemplo para o aprendizado por reforço pode ser a atribuição de medias, a determinado elemento, onde pode se tentar inibir entradas de grande disparidade, para manter os resultados em determinada harmonia. Quando se quer saber a variação de preços de determinados produtos, pode se excluir o preço das liquidações, por ocorrerem em momentos isolados.

2.3.4 Algoritmo de Aprendizado

Existem vários algoritmos de treinamento para RNAs, com suas vantagens e desvantagens. Para este trabalho foi escolhido o algoritmo de *backpropagation* que, segundo Braga (2000, p. 59) é um dos mais conhecidos em treinamentos de RNAs, sendo um algoritmo supervisionado, o que o torna mais vantajoso quando se utiliza um especialista do domínio.

O algoritmo de treinamento *backpropagation* baseia-se na retropropagação, ajustando os pesos das unidades das camadas intermediarias, através dos erros identificados na camada de saída (FARACO, 1998, p. 28-29). Segundo Barca (2005, p.47) o processo de treinamento

do algoritmo é baseado em tentativa e erro, o que reflete diretamente no tempo de treinamento, o qual consome muito tempo de processamento.

Segundo Severo (2010, p. 14), o Algoritmo Pseudocódigo do *Backpropagation* apresenta as seguintes etapas:

1. inicializar pesos e bias;
2. apresentar o padrão de entrada juntamente com sua respectiva saída desejada;
3. propagar esse padrão de camada em camada de forma que seja calculada a saída para cada nodo da rede
4. comparar a saída gerada pela rede com a saída desejada e calcular o erro cometido pela rede para os nodos da camada de saída
5. atualizar os pesos dos nodos da camada de saída com base no erro competido por tais nodos
6. até chegar à camada de entrada:
    a. calcular o erro dos nodos da camada intermediária baseado no erro cometido pelos nodos imediatamente seguintes ponderado pelos pesos entre os nodos da camada atual e os nodos imediatamente seguintes
7. Repetir os passos 2, 3, 4, 5 e 6 até obter um erro mínimo ou até atingir um dado número de iterações

2.4 Fisioterapia

Conforme o Conselho Federal de Fisioterapia e Terapia Ocupacional (COFFITO, 2012, *online*), a Fisioterapia é definida como uma ciência da Saúde que estuda, previne e trata os distúrbios cinéticos funcionais intercorrentes em órgãos e sistemas do corpo humano, gerados por alterações genéticas, por traumas e por doenças adquiridas. Fundamenta suas ações em mecanismos terapêuticos próprios, sistematizados pelos estudos da Biologia, das ciências morfológicas, das ciências fisiológicas, das patologias, da bioquímica, da biofísica, da biomecânica, da cinesia, da sinergia funcional, e da cinesia patológica de órgãos e sistemas do corpo humano e as disciplinas comportamentais e sociais.

E que, em suas atribuições, os profissionais da fisioterapia podem elaborar o Diagnóstico Cinesiológico Funcional, prescrever, planejar, ordenar, analisar, supervisionar e avaliar os projetos fisioterapêuticos, a sua eficácia, a sua resolutividade e as condições de alta do cliente submetido a estas práticas de saúde. Dentro da Fisioterapia existe a área da fisioterapia traumato-ortopedia, que será apresentada na próxima seção.

2.4.1 Fisioterapia Traumato-Ortopédica

A Fisioterapia Traumato-Ortopédica é reconhecida como uma Especialidade própria e exclusiva do Fisioterapeuta é uma especialidade que busca corrigir problemas relacionados a contusões, traumas, lesões o que impossibilita ou dificulta a movimentação correta do corpo humano, suas patologias são dividas em regiões do corpo, estas regiões são ilustradas na figura 15 (BRASIL, 2004).

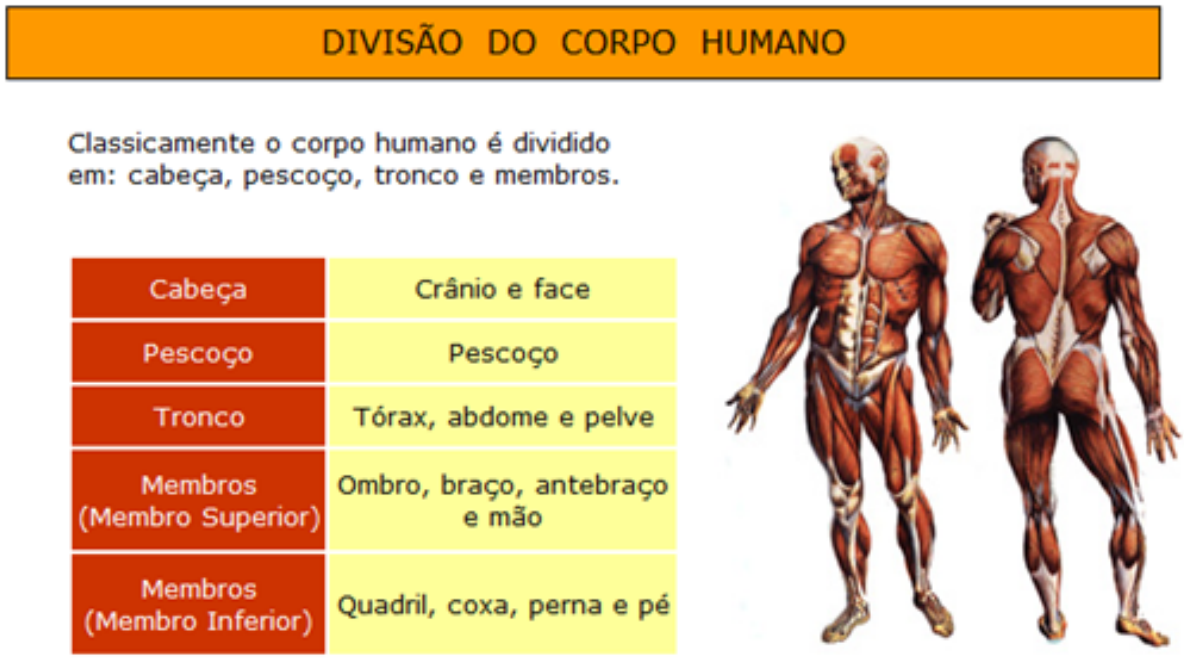

Figura 16 - Divisão do corpo humano, adaptado de Borba (2011).

Na figura 16, pode ser observada a divisão das partes do corpo humano, sendo que, em específico para este trabalho, serão analisados os membros inferiores, em especial a região do quadril, coxa, joelho e pé. As patologias traumato-ortopédicas dos membros inferiores caracterizam-se por sinais e sintomas originados nos membros inferiores do corpo humano, como: quadril e coxa, joelho, tornozelo e pé. A tabela contendo as patologias e seus sinais e sintomas completos estão no **Apêndice A**.

3 MATERIAIS E MÉTODOS

Nesta seção serão apresentados os materiais e métodos utilizados no desenvolvimento do trabalho.

3.1. Contato com o especialista

Para o desenvolvimento desse trabalho foi necessária a colaboração de um profissional da área da fisioterapia ortopédica, para especificar o domínio de aplicação, determinar os dados a serem utilizados e validar os resultados obtidos. Então, contou-se com a disponibilidade do professor Pierre Brandão, fisioterapeuta, mestre em gerontologia e coordenador do laboratório de Instrumentalização Científica do Centro Universitário Luterano de Palmas - CEULP/ ULBRA. Na próxima seção será abordada a metodologia e apresentados os passos utilizados para o desenvolvimento do trabalho.

3.2. Metodologia

A primeira etapa da metodologia foi entrar em contato com um especialista do domínio da aplicação, que conhecesse as peculiaridades das patologias traumato-ortopédica dos membros inferiores e que pudesse indicar quais seriam utilizadas no trabalho, como também o material para pesquisa destas patologias.

A segunda etapa foi caracterizada pela coleta de informação a respeito do domínio do trabalho, que envolve Inteligência Artificial (IA), Redes Neurais Artificiais (RNAs), além de informações sobre a ferramenta a ser utilizada, o JavaNNS. Sendo assim, foi realizada a busca de trabalhos científicos, artigos, teses, dissertações, entre outros, em meios físicos ou através da internet, como também a busca de livros com este conteúdo.

Já a terceira etapa constituiu o estudo das informações coletadas na segunda etapa, e a criação de fichamentos contendo citações dos autores dos trabalhos, como também a busca por ilustrações que melhorem a compreensão das mesmas, exemplificando as informações expostas.

A quarta etapa foi caracterizada pela escrita da revisão de literatura deste trabalho, utilizando as informações que foram obtidas na etapa anterior, como também as ilustrações e tabelas. A primeira parte desta etapa diz respeito a IA, descrevendo seus conceito e suas características. Na segunda parte foi feita a descrição das Redes Neurais, iniciando pela biológica e posteriormente a artificial, expondo uma comparação entre ambas para uma

melhor compreensão do conteúdo. Na terceira parte foi feito um estudo da ferramenta de treinamento da RNA, o JavaNNS, cuja descrição e funcionalidades serão apresentadas a seguir. Por fim, a quarta parte foi caracterizada pela escrita sobre as patologias ortopédicas dos membros inferiores, destacando suas características, além das tabelas contendo as relações de seus sinais e sintomas.

A importância em seguir as etapas mencionadas anteriormente está diretamente ligada a segunda fase do trabalho, que utilizará os conceitos apresentados, como também a necessidade do especialista em sua concepção, pois sem estas informações não há uma compreensão de como o sistema será criado, no que diz respeito a seus formato e características.

Depois de feitas as etapas anteriores, foi iniciada a fase de desenvolvimento do sistema, conforme pode ser observado nos seguintes passos:

- criação de tabelas com as relações entre as patologias e seus sinais e sintomas, como também tabela contendo só as patologias e outra com os sinais e sintomas, esta divisão ocorreu pela descrição da representação deste elementos na RNA, como também pela eliminação de repetições de sinais e sintomas;
- com as tabelas criadas, iniciou-se a parte de pré-processamento, que ocorreu na tabela de sinais e sintomas, em que os sinais e sintomas que se repetiam várias vezes foram eliminados deixando somente uma ocorrência. Logo, foram criadas representações para estes dados na rede, atribuindo valores dentro do conjunto de 0 a 1 para as tabelas de patologias e de sinais e sintomas, criando assim novas tabelas relacionando os sinais e sintomas e as patologias com suas respectivas representações para a RNA;
- instalação e configuração do JavaNNS: neste passo foi feita a instalação dos componentes do JavaNNS e as configurações para seu funcionamento, como a instalação do ambiente Java na máquina onde é executado o JavaNNS, utilizando o *Java Runtime Environment* (JRE) versão 6. Depois da instalação e configuração foi feita uma leitura da documentação da ferramenta em que são especificadas suas funções e como ela é utilizada. O JavaNNS fornece exemplos prontos com estruturas de redes neurais e arquivos de treinamento para que, ao se utilizar estes exemplo, seja possível aprender a utilizar suas funções;
- definições da RNA: neste passo foram feitas as definições da rede neural, como a quantidade de neurônios da camada de entrada, escondida, e da saída. Observando que poderiam haver várias estruturas para uma rede neural, foi especificada a criação

de quatro redes neurais com estruturas distintas, para identificar qual estrutura teria melhor eficiência, porém não alterando a quantidade de neurônio de entrada e nem de saída, pois o arquivo de padrões tem esse tipo de especificação. Sendo assim foram definidas redes neurais com dez entradas, pois o máximo de sinais e sintomas identificado nas tabelas feitas anteriormente aponta dez sinais e sintomas para uma patologia. Como o arquivo de padrões não aceita uma entrada em branco, foi definido que para um sinal ou sintoma ausente será atribuído o valor zero, e para a saída um neurônio que representa as patologias, o que sofrerá modificações será a camada oculta da rede neural;

- criação dos arquivos de padrões: depois de ter definido o formato da rede neural foram criados os arquivos de padrões. São formados por dois arquivos, um contendo os padrões de treinamento, e outro com os padrões de validação da rede. A divisão ocorre, pois a rede deve ser treinada com um conjunto de dados distinto do conjunto utilizado para validação. A diferença entre eles é que os padrões de validação são desconhecidos pela rede, a rede verifica se, de acordo com o que foi treinado, consegue atingir os resultados esperados nos dados de validação, gerando assim a generalização da rede. Ambos os arquivo de padrões são feito manualmente utilizando os sinais e sintomas e as suas referidas patologias, contudo utilizando a sua representação numérica para a rede neural, os arquivos de padrões utilizados neste trabalho podem ser vistos nos apêndices C e D;
- treinamento da RNA: para o treinamento da rede neural foram estabelecidas várias definições que serão apresentadas posteriormente. No treinamento da rede neural foi utilizado o arquivo de padrões de treinamento como mencionado anteriormente, selecionado o algoritmo de treinamento, o *Backpropagation*;
- coleta de resultados: o JavaNNS oferece alguns tipos de visualizações dos resultados do treinamento como um gráfico de Erro Médio Quadrático por Ciclos de Treinamento, que apresenta a evolução da rede pelo período de treinamento e tabela de *logs* da rede, com estas informações foi iniciada a coleta de resultados da rede neural;
- análise dos dados: depois de ter coletado os dados das quatro redes neurais foi iniciada a análise que tem como objetivo apresentar os resultados e comparar com as demais RNAs para verificar quais obtiveram melhor performance e determinar quais destas é a melhor solução para o domínio da aplicação;

- validação dos Resultados: depois de obter a análise das redes neurais, os dados de performance foram apresentados ao especialista do domínio para validação, sendo que o mesmo considerou que as redes foram eficientes no diagnóstico das patologias traumato-ortopédicas dos membros inferiores, através de testes realizados com a rede neural.

Depois de finalizar a análise dos resultados foi necessário revisar o trabalho, corrigindo e acrescentando alguns detalhes e conteúdo que foram surgindo com a concepção da rede, em geral algumas definições que não foram abrangidas na parte da revisão de literatura inicial.

3.3. Escolha da ferramenta

Existem várias ferramentas que simulam as redes neurais, Souza Filho (2009, p. 53-54) faz uma pesquisa onde lista e compara seis ambientes de criação de redes neurais:

- Java Neural Network Simulator (JavaNNS)
- Stuttgart Neural Network Simulator (SNNS)
- Java Object Neural Engine (Joone)
- Artificial Neural Networks Framework (ANNeF)
- Fast Artificial Neural Network Library (FANN)
- Scilab

Pode ser observada na tabela 1 a comparação entre os ambientes de criação de RNAs relacionando os recursos oferecidos por cada ambiente, a simplicidade de uso e seu reconhecimento na área acadêmica.

Tabela 1: Análise e Pontuação dos Ambientes de Criação de RNAs (SOUZA FILHO, 2009, p. 54).

Ambiente	**Recursos Oferecidos**	**Simplicidade de Uso**	**Reconhecimento Acadêmico**	**Total**
	(1 a 3)	**(1 a 3)**	**(1 a 3)**	
JavaNNS	3	3	3	**9**
SNNS	3	2	3	**8**
JOONE	2	2	2	**6**
ANNeF	2	1	2	**5**
FANN	2	1	2	**5**
Scilab	2	1	3	**6**

Na tabela 1 pode ser observado que os ambientes são comparados pelos recursos oferecidos como, por exemplo, se o ambiente fornece opções para criação, treinamento e teste de RNAs, como também a simplicidade de uso verificando a facilidade de realizar os

procedimentos como os citados anteriormente e, por fim, o reconhecimento acadêmico, em que é analisado se a ferramenta é recomendada por instituições de ensino e se é bastante utilizada em trabalhos acadêmicos.

Assim, verificando as informações da Tabela 1, percebe-se que o JavaNNS teve maior pontuação sobre as demais, além de obter nota máxima em todos os quesitos. Por atender as necessidades deste trabalho, foi escolhido utilizar o JavaNNS como ambiente de criação de RNAs. Na próxima seção é feita a apresentação mais detalhada da ferramenta.

3.4. *Java Neural Network Simulator* - JavaNNS

O *Java Neural Network Simulator* (JavaNNS) é um simulador de redes neurais desenvolvido no *Wilhelm-Schickard-Institute for Computer Science* (WSI) em Tübingen, na Alemanha e é uma versão em Java do *Stuttgart Neural Network Simulator* (SNNS) que foi desenvolvido pelo *Institute for ParallelandDistributed High Performance Systems* (IPVR) da Universidade de Stuttgart. É uma ferramenta gratuita em que é possível projetar e treinar uma rede neural.

O JavaNNS pode ser utilizado nos seguintes sistemas operacionais: Linux Ubuntu, Windows XP, Windows 7. Na figura 17 é observado o JavaNNS em execução na sua fase de treinamento de uma rede neural e as informações apresentadas por esta.

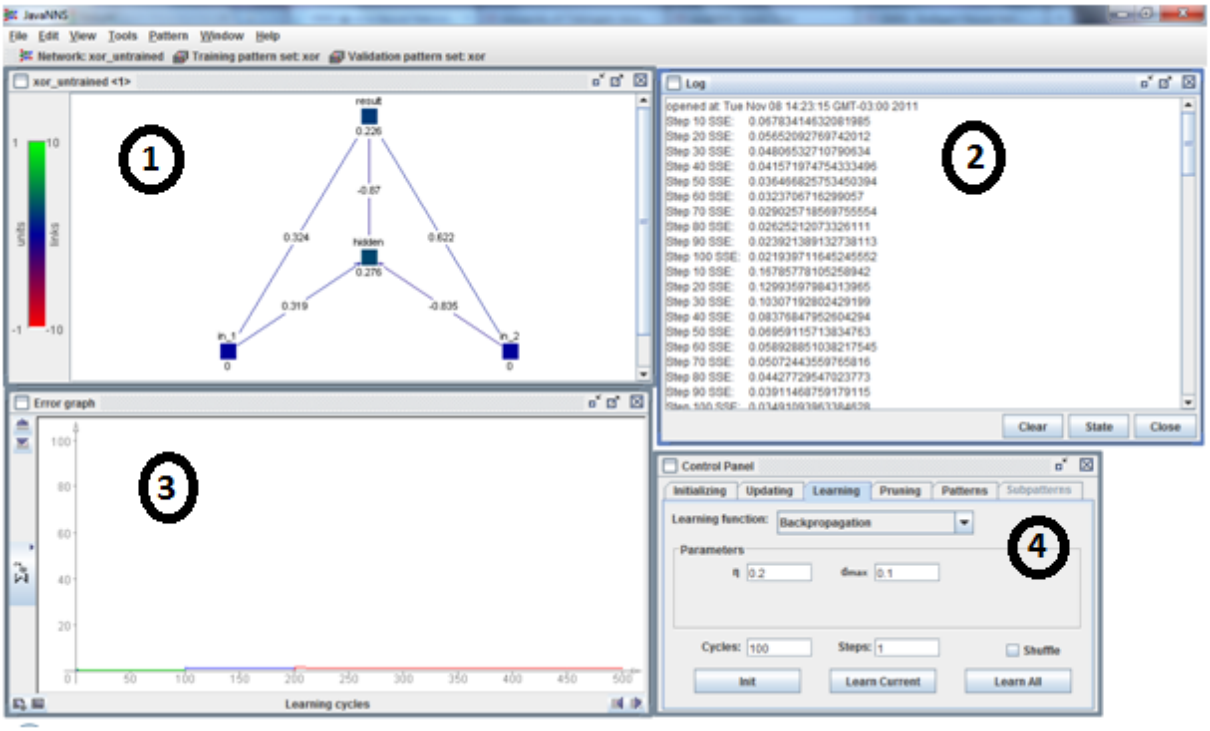

Figura 17 - Componentes do JavaNNS.

Na figura 17 é ilustrado uma rede neural em treinamento utilizando o JavaNNS. Na parte marcada com o número 1 é observada a visualização gráfica da RNA; na parte marcada com o número 2, os registros de *log* que são gravados a cada ciclo de treinamento; na parte marcada com o número 3, o gráfico de erro da rede; na parte marcada com o número 4 o painel de controle, onde é possível selecionar o algoritmo de treinamento, bem como definir os parâmetros da rede, como as opções de iniciar o treinamento da mesma.

O JavaNNS trabalha com diferentes tipos de arquivos utilizados no ambiente da ferramenta:

- .net arquivos de rede (unidades e ligações) armazena a estrutura da rede que inclui as camadas e ligações;
- .pat arquivos de padrões utilizado para treinamento e validação ou testes da rede;
- .cfg arquivos de configurações;
- .txt arquivos de texto (log files);
- .res arquivos de resultados (ativações de unidades).

Nas próximas seções será apresentado como são realizados a criação e o treinamento de uma Rede Neural Artificial utilizando o JavaNNS.

3.5. Criação da Rede Neural

Para a criação da RNA será utilizado o JavaNNS, para isso, o primeiro passo é a criação das camadas. São observados na figura 18 os primeiros passos para a criação das camadas ou *Layers*.

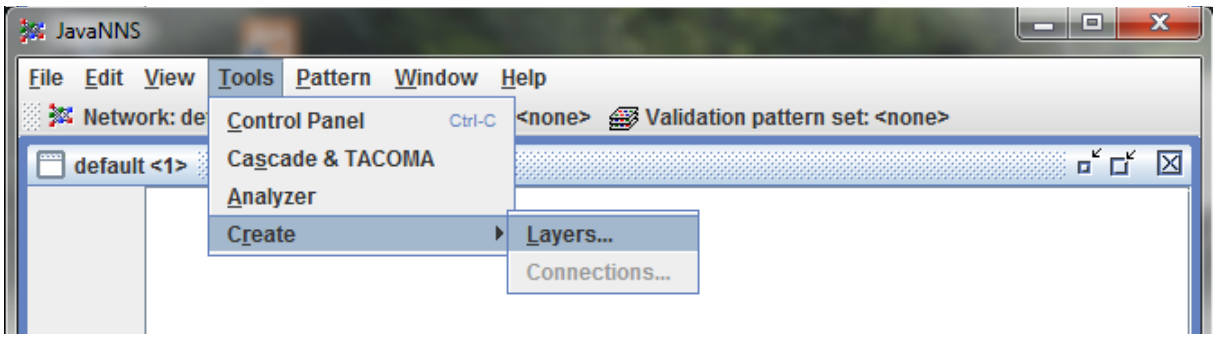

Figura 18 - Primeiro passo para criação das camadas da rede.

Na figura 18 pode ser observado que, para iniciar a criação das camadas, depois da ferramenta iniciada, é necessário selecionar a opção *Create->Layers* dentro do menu *Tools*, e depois de selecionar esta opção uma janela de diálogo irá surgir na tela, como observado na

figura 19.

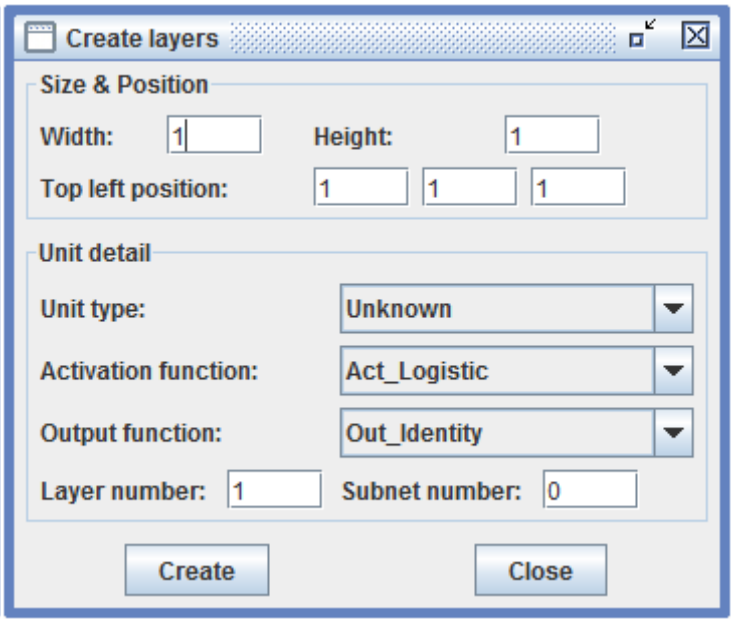

Figura 19 - Caixa de diálogo para criação das camadas.

De acordo com Souza Filho (2009, p. 73) a caixa de diálogo que pode ser observada na figura 19 solicita informações para a criação das camadas da rede neural. Seus campos possuem os seguintes significados:

- *Width*: quantidade de neurônios a serem criados na posição horizontal;
- *Height*: quantidade de neurônios a serem criados na posição vertical;
- *Top left position*: posição inicial na qual devem ser criados os neurônios;
- *Unit type*: define o tipo de neurônio a ser criado, pode-se escolher entre a camada de entrada (*input*), a camada oculta (*hidden*), a camada de saída (*output*), entre outras.
- *Output function*: define qual será a função de saída a ser utilizada.
- *Layer number*: número da camada a ser criada;
- *Subnet number*: número da rede interna a ser criada.

Para a construção da rede neural, serão configurados os seguintes parâmetros: dez neurônios na camada de entrada, um na camada escondida, e um na camada de saída, que no final da criação possui a aparência observada na figura 20.

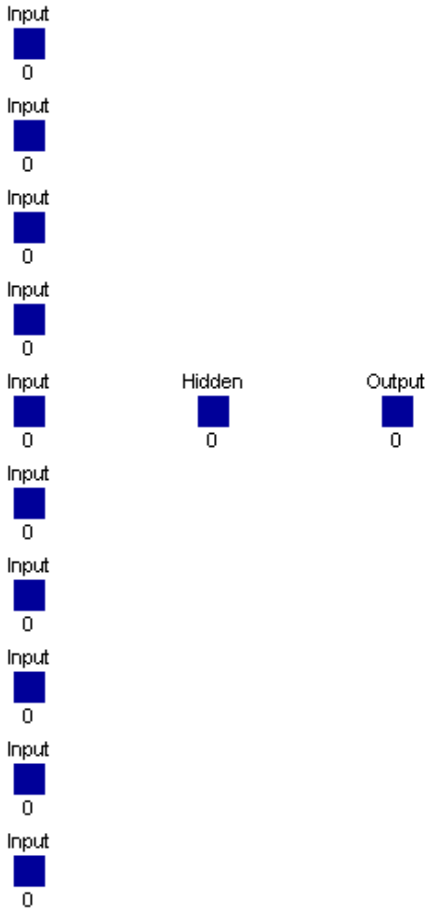

Figura 20 - Camadas da rede neural.

Depois de ter criado as camadas com os neurônios, como observado na figura 20, é necessário criar as conexões entre estes neurônios. No menu *Tools->Create* há a opção *Connections* que é onde inicia a criação destas conexões, como pode ser observado na figura 21.

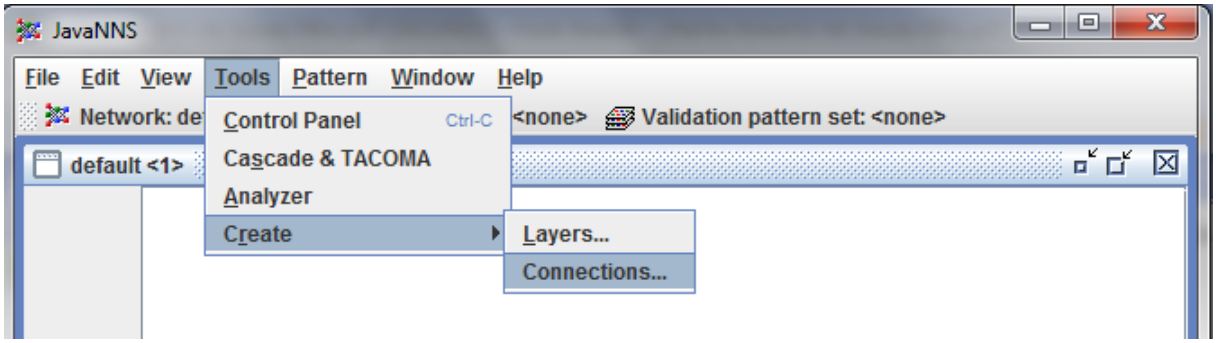

Figura 21 - Opção *Connections*.

Na figura 21 pode-se verificar que, depois de selecionar a opção Connections, será apresentada uma caixa de diálogo onde são oferecidas as opções de conexões entre as camadas da rede neural, como é observado na figura 22.

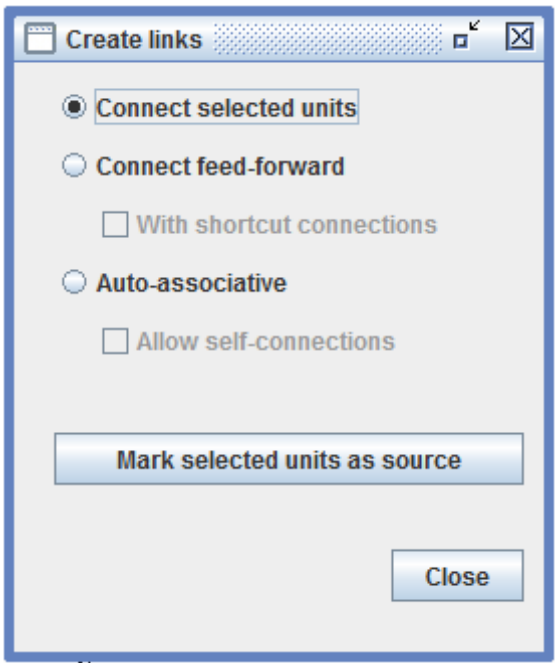

Figura 22 - Caixa de diálogo para criar as conexões da RNA.

Na figura 22 a caixa de diálogo apresenta algumas opções para a criação das conexões das camadas da RNA, e segundo Souza Filho (2009, pag. 73) estas opções têm o seguinte significado:

- *Connect selected units*: permite selecionar os neurônios desejados e criar as conexões manualmente;
- *Connect feed-forward*: cria conexões do tipo *feed-forward*;
- *Auto-associative*: cria conexões auto-associativas.

Conforme as definições do trabalho, a arquitetura de rede que será utilizada no trabalho será a *feed-forward*, por ser uma arquitetura de rede supervisionada, e por isso na caixa de diálogo observada na figura 22, é selecionada a opção referente a esta arquitetura, para se obter uma estrutura de RNA observada na figura 23.

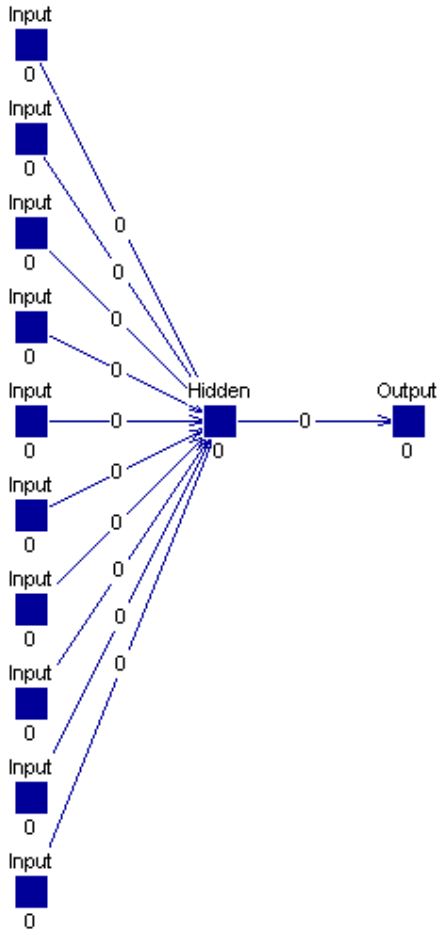

Figura 23 - RNA com arquitetura *feed-forward*.

Na figura 23 pode ser observada a estrutura da RNA depois de finalizada a sua construção, onde se pode perceber que existem cinco elementos de entrada, uma camada oculta e um elemento que representa a saída da rede, todas as camadas ligadas formando uma arquitetura *feed-forward.* Depois da rede criada, inicia-se a etapa de treinamento da rede, apresentada na próxima seção.

3.6. Treinamento da Rede Neural

Para iniciar o treinamento da RNA com o JavaNNS, é preciso criar um arquivo com os parâmetros de treinamento em um formato válido para o JavaNNS. Para isso é necessário que seja criado um arquivo de extensão .pat com as definições observadas na figura 24.

```
1  SNNS pattern definition file V3.2
2  generated at Mon Apr 25 15:58:23 1994
3
4
5  No. of patterns : 6
6  No. of input units : 5
7  No. of output units : 1
8
9  # Input pattern 1:
10 0.04 0.05 0.06 0 0
11 # Output pattern 1:
12 0.4
13 # Input pattern 2:
14 0.07 0.08 0.09 0 0
15 # Output pattern 2:
16 0.5
```

Figura 24 - Arquivo de treinamento do JavaNNS.

Na figura 24 pode ser observada a estrutura do arquivo utilizado para o treinamento da rede neural: o cabeçalho nas linhas 1 e 2 são padrões JavaNNS para reconhecer o arquivo; nas linhas 5 a 7 são as definições do arquivo de treinamento informando a quantidade de padrões de treinamento, neurônios de entrada e neurônios de saída; nas linhas posteriores seguem os padrões para treinamento, lembrando que as linhas iniciadas com # (cerquilha) são comentários do arquivo.

Depois de criar o arquivo de treinamento, é necessário executá-lo e, para isso, o JavaNNS oferece um painel de controle destas operações dentro do menu *Tools* a opção *Control Panel*, como observado na figura 25.

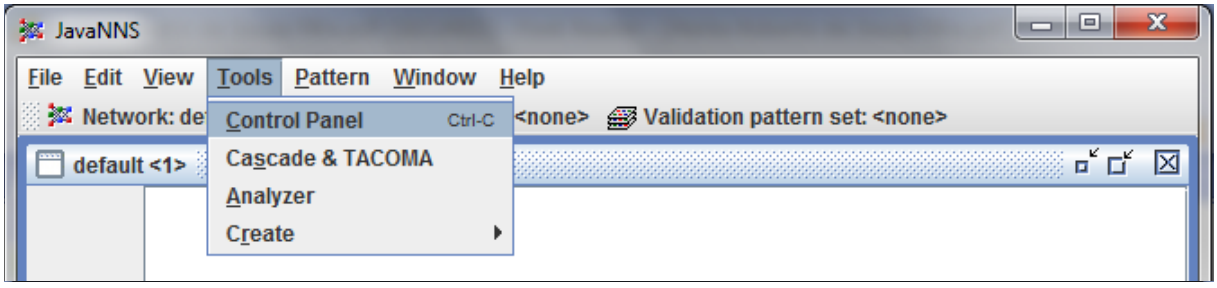

Figura 25 - Opção para iniciar o *Control Panel*.

Depois de selecionar a opção *Control Panel*, como apresentado na figura 25, é aberta uma caixa de diálogo com várias abas que contém os comandos e configurações para o treinamento da rede (figura 26).

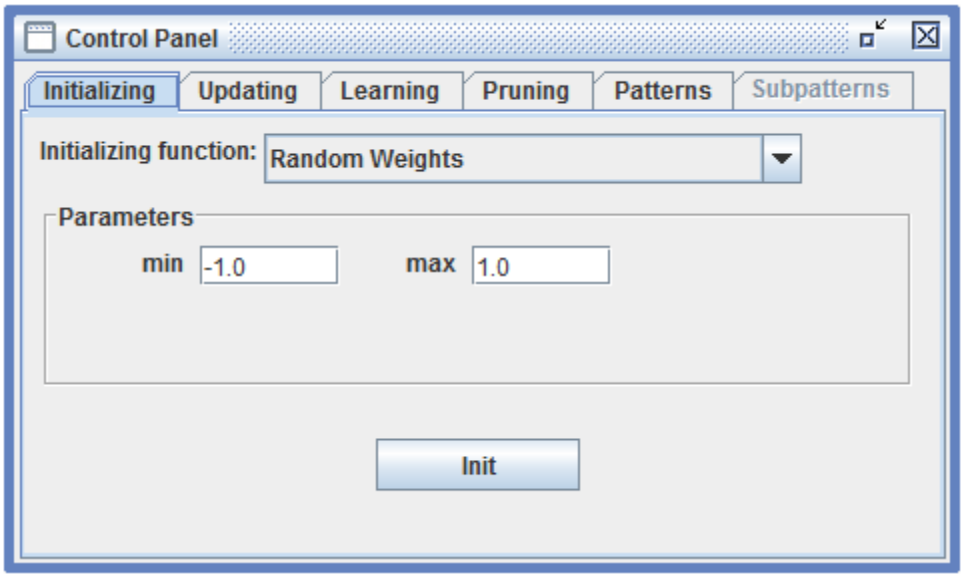

Figura 26 - Caixa de diálogo *Control Panel* aba *Initializing*.

Na figura 26, pode-se ver a primeira aba do *Control Panel* que traz a configuração de *Initializing*, que é onde se inicializam os pesos da RNA de forma aleatória. A função de inicialização deve ser escolhida em *inicializing function*. Feito isso, basta definir os parâmetros da função escolhida e posteriormente selecionar a opção *Init*. Na próxima aba do *Control Panel* existe a opção de Updating observado na figura 27.

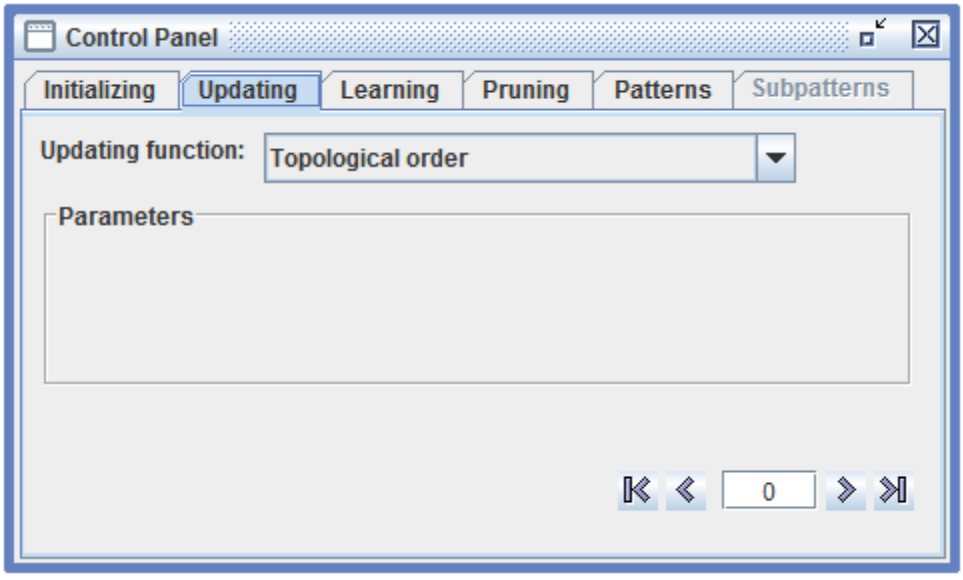

Figura 27 - Caixa de diálogo *Control Panel* aba *Updating*.

Na aba *Updating,* (figura 27), é apresentada a opção *Updating function,* em que é feita a escolha da função de atualização utilizada na RNA, onde é utilizada a função *Topological order*. Dependendo da função escolhida, é preciso definir parâmetros. A próxima aba traz as opções de *Learning*, como apresentado na figura 28.

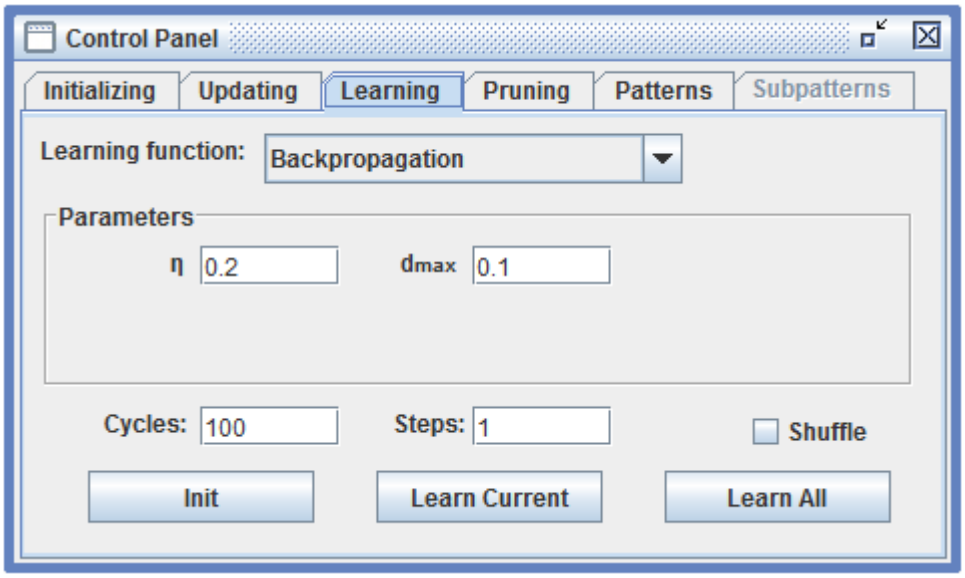

Figura 28 - Caixa de diálogo *Control Panel* aba *Learning*.

Na figura 28 podem ser observados que na aba *Learning* são oferecidas diversas opções para o aprendizado da rede. A função de aprendizado deve ser escolhida em *Learning function* e, depois disto, basta definir os parâmetros da função escolhida. Existem também outros dois campos: o *Cicles*, que especifica o número de ciclos de aprendizagem e o *Steps* que especifica o número de passos de atualização da rede. Há também um *checkbox* denominado *Shuffle* que, se estiver selecionado, define que os padrões de aprendizado serão lidos aleatoriamente, caso contrário serão lidos ordenadamente. Por fim existem três botões na parte inferior da caixa de diálogo que têm os seguintes significados:

- *Init*: inicializa a rede da mesma forma como ocorre na aba *Initializing*;
- *Learn Current*: treina a rede utilizando apenas uma entrada do arquivo de padrões; e
- *Learn All*: treina a rede utilizando todas as entradas contidas no arquivo de padrões.

Na próxima aba observada na figura 29 são apresentadas as opções de poda da RNA.

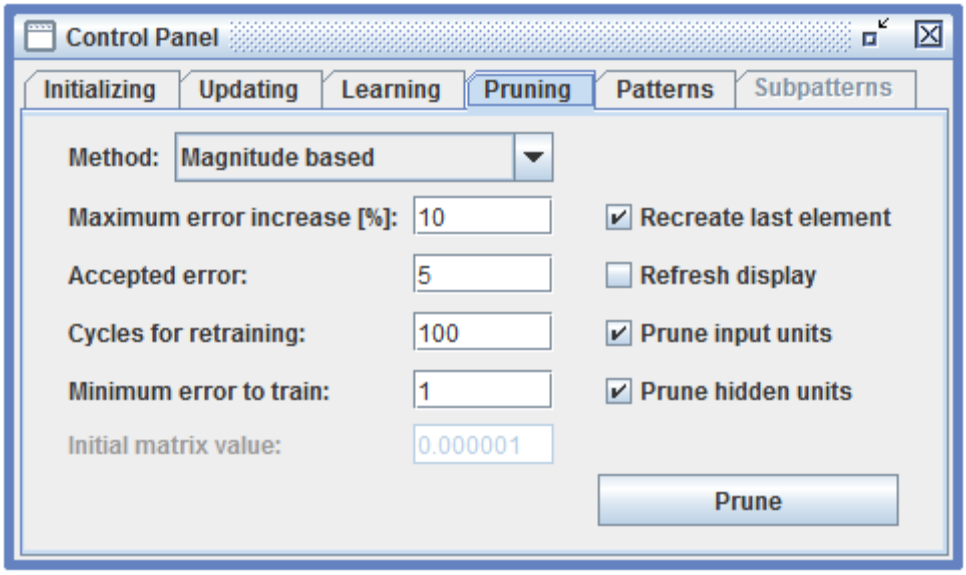

Figura 29 - Caixa de diálogo *Control Panel* aba *Pruning*.

Na figura 29 podem ser vistas as opções da aba *Pruning*. O método de poda da rede deve ser definido no campo *Method*. Feito isso é preciso definir os parâmetros do método escolhido e então aplicar a opção Prume, porém esta opção não será utilizada neste trabalho, por não haver a necessidade de seu uso, pois a rede em especifico tem limites de treinamento. Na figura 30 pode ser visualizada a próxima aba denominada *Patterns*.

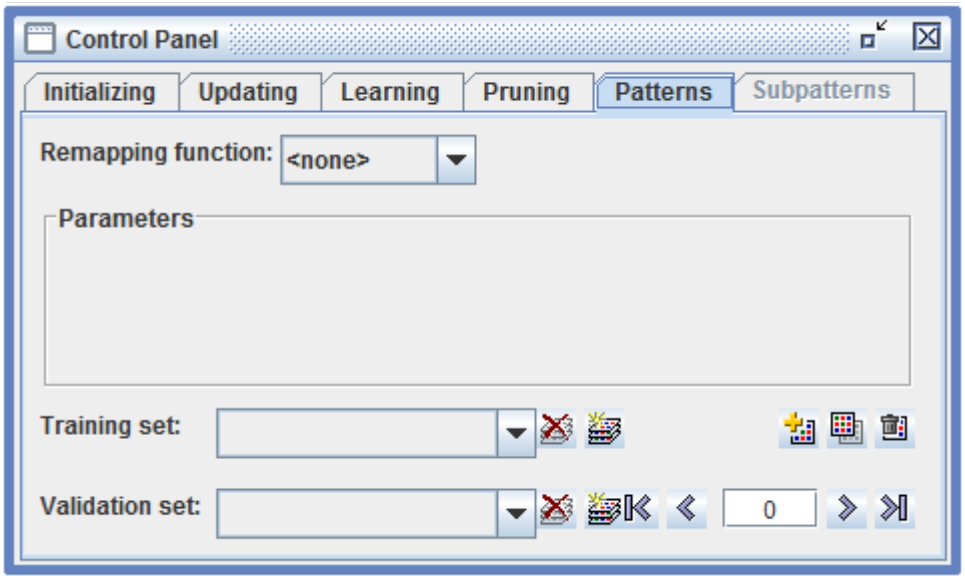

Figura 30 - Caixa de diálogo *Control Panel* aba *Patterns*.

Na figura 30 a aba *Patterns* do *Control Panel* define os conjuntos ou arquivos de entrada e saída da RNA. Trata-se dos arquivos de padrões de treinamento e validação. A primeira opção que esta aba fornece é a *Remmapping function* ou função de re-mapeamento. Feita a escolha da função é necessário definir seus parâmetros. Logo abaixo existem os campos *Training set,* em que é escolhido o arquivo de treinamento a ser utilizado pela RNA, e o *Validation set*, que define o arquivo de validação da RNA.

Para poder selecionar um arquivo no *Training set e Validation set* é necessário inicialmente que os mesmos estejam carregados no sistema. Para carregá-los, seleciona-se a opção *File* e depois a opção *Open*, selecionando arquivos da extensão .pat. Existe também a aba *Subpatterns* que apresenta os parâmetros referentes aos subconjuntos de entrada e saída, quando utilizados. Por fim, na figura 31 pode ser observada a ilustração da rede treinada.

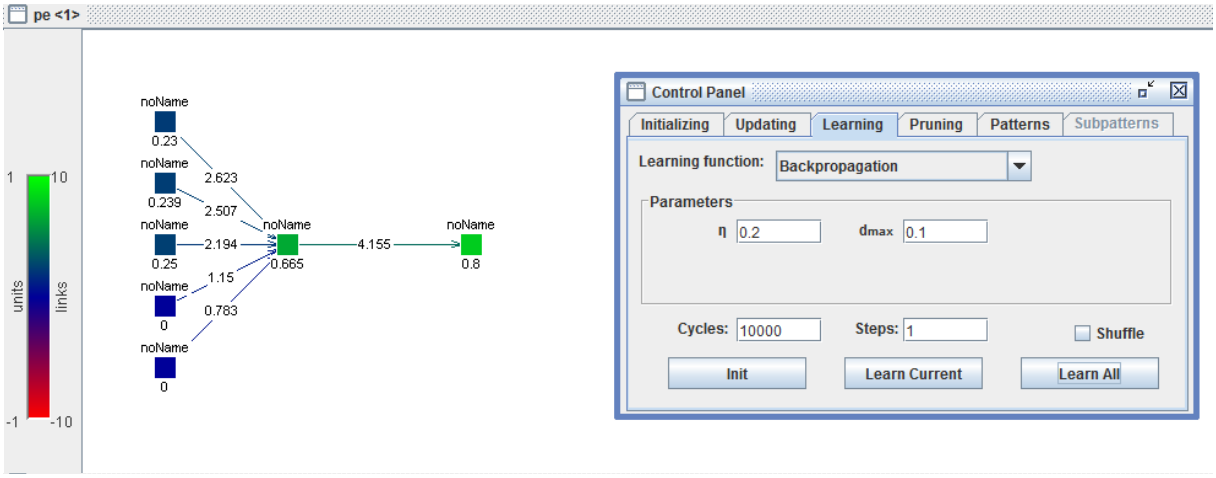

Figura 31 - Ilustração da RNA treinada.

Pode-se observar que o JavaNNS possui uma interface amigável, o que facilita na identificação e utilização de suas funções, fornecendo visibilidades nas etapas de criação, treinamento e validação da rede. Na próxima seção serão apresentados os resultados e discussões do trabalho.

4 RESULTADOS E DISCUSSÃO

Nesta seção serão analisados os resultados obtidos com a criação, treinamento e validação da rede neural, como também a comparação de quatro redes, utilizando os mesmos atributos, porém com pequenas modificações em suas estruturas, em que se busca determinar uma rede com desempenho melhor.

Além disso, será apresentada a validação dos resultados da aplicação de redes neurais artificiais para o diagnóstico de patologias traumato-ortopédicas dos membros inferiores pelo especialista do domínio.

4.1. Dados e informações

Os dados e informações utilizados neste trabalho foram pesquisados e obtidos através da orientação do especialista do domínio. Trabalhou-se com as patologias traumato-ortopédica dos membros inferiores e, baseado nas informações obtidas com o especialista, foram construídas as tabelas que se encontram no **Apêndice A**, que descrevem os sinais e sintomas relacionados a cada patologia.

Ao analisar estas tabelas pode-se observar que várias patologias possuem sinais e sintomas semelhantes, como, por exemplo, as patologias Meniscectomia, Fraturas de estresse e Fraturas do tornozelo têm a dor como sinal e sintoma comum, e a mesma patologia Meniscectomia e a Tríade infeliz (O'Donoghue) têm atrofia como um sinal e sintoma semelhante. Por existirem estas ocorrências, é preciso identificar quais sinais e sintomas que se repetem nas outras patologias, e que têm as mesmas características das demais. Assim, estes dados necessitam ser pré-processados e normalizados, sendo atribuídos a estes, valores representativos na RNA, que serão vistos na próxima seção.

4.2. Pré-processamento e Normalização

Antes de utilizar os dados na RNA, estes precisam ser pré-processados e posteriormente normalizados, para que sejam eliminadas as redundâncias e criada a representação para a RNA. O pré-processamento constitui na limpeza dos dados que serão utilizados na RNA, pois nesta etapa os dados que serão utilizados, no caso os sinais e sintomas, serão agrupados. É feita a organização e a eliminação de ocorrência redundante. Esta eliminação constitui-se da

junção das ocorrências de sinais e sintomas em um único elemento, evitando a redundância destes dados.

Na normalização dos dados será definido um valor para a sua representação na RNA. Na rede neural serão utilizados valores entre 0 e 1 para representar os sinais e sintomas que podem ser vistos no **Apêndice B**, e a representação para as patologias. Na tabela 1 podem ser observadas as patologias com as saídas que serão representadas na rede neural.

Tabela 2: Relação das patologias com suas respectivas representações na RNA.

Descrição da patologia	Representação na RNA
Doença articular degenerativa	0,1
Distensões e lacerações musculares	0,11
Bursite trocantérica	0,12
Pontada no quadril	0,13
Meralgiaparestésica	0,14
Miositeossificante	0,15
Doença de Legg-Calvé-Perthes	0,16
Deslocamento da epífese da cabeça do fêmur	0,17
Luxação congênita do quadril	0,18
Artroplastia total do quadril	0,19
Entorse/laceração do ligamento colateral medial	0,2
Entorse/laceração do ligamento colateral lateral	0,21
Entorse (deficiência) do LCA	0,22
Entorse do LCP	0,23
Tendinite quadricipital ou patelar (joelho do saltador)	0,24
Ruptura do tendão quadricipital ou patelar	0,25
Bursite patelar	0,26
Bursite anserina (da pata de ganso)	0,27
Síndrome da plica	0,28
Síndrome do atrito do trato iliotibial (TIT)	0,29
Lacerações meniscais	0,3
Tríade infeliz (O'Donoghue)	0,31
Osteoartrite	0,32
Osteocondrite dissecante	0,33
Meniscectomia	0,34
Síndrome de dor patelofemoral	0,35
Luxação/subluxação patelar	0,36
Entorse/lacerações agudas do tornozelo lateral	0,37
Instabilidade crônica do tornozelo lateral	0,38
Doença de Sever	0,39
Entorses sindesmóticas (diástase do tornozelo)	0,4
Tendinite do tibial posterior	0,41
Tendinite do Aquileu	0,42
Ruptura do tendão-de-Aquiles	0,43
Lacerações do gastrocnêmio	0,44

Síndrome compartimental por esforço crônico	0,45
Síndrome compartimental aguda	0,46
Fraturas de estresse	0,47
Fraturas do tornozelo	0,48
Reparo no tendão de Aquiles	0,49
Reconstrução dos ligamentos do tornozelo	0,5
Fasciite plantar	0,51
Síndrome do túnel do Tarso	0,52
Neuroma de Morton	0,53
Artelho da grama artificial	0,54
Pronação anormal	0,55
Supinação anormal	0,56

Analisando as tabelas com sinais e sintomas e as patologias, observa-se que cada um possui uma representação única na rede neural. Isso se deve ao fato que a rede neural deve interpretar cada dado de forma única como em um caso real e, quando houver a necessidade de utilização de um mesmo dado, repetidas vezes, compreender que aquele dado representa o mesmo valor que em outras ocorrências. Por exemplo, o sintoma dor repete-se várias vezes e na rede neural é representado pela sequência numérica 0,169. A rede interpreta que cada ocorrência deste valor tem o mesmo significado.

Depois que os dados estão pré-processados, normalizados e possuem sua representação para a rede neural, é iniciada a criação do arquivo de padrões, utilizado para o treinamento e validação da rede, que será apresentado na próxima seção.

4.3. Arquivo de Padrões

O arquivo de padrões contém os dados para treinamento da rede neural. No JavaNNS são utilizados dois arquivos de padrões: um com 35 padrões utilizados para o treinamento da rede neural e outro com 12 padrões utilizados para sua validação, onde a quantidade de total utilizado no trabalho é de 47 padrões. O arquivo de padrões contém a descrição dos sinais e sintomas utilizando sua representação para a RNA, como também sua respectiva saída. O JavaNNS possui um formato próprio de arquivo de padrões, o qual é criado manualmente, sendo formado por um conjunto de dados de entrada e sua saída ideal. Na figura 32 pode ser observada parte do arquivo de padrões utilizado para o treinamento da rede. O arquivo de padrões completo pode ser observado no **Apêndice C**.

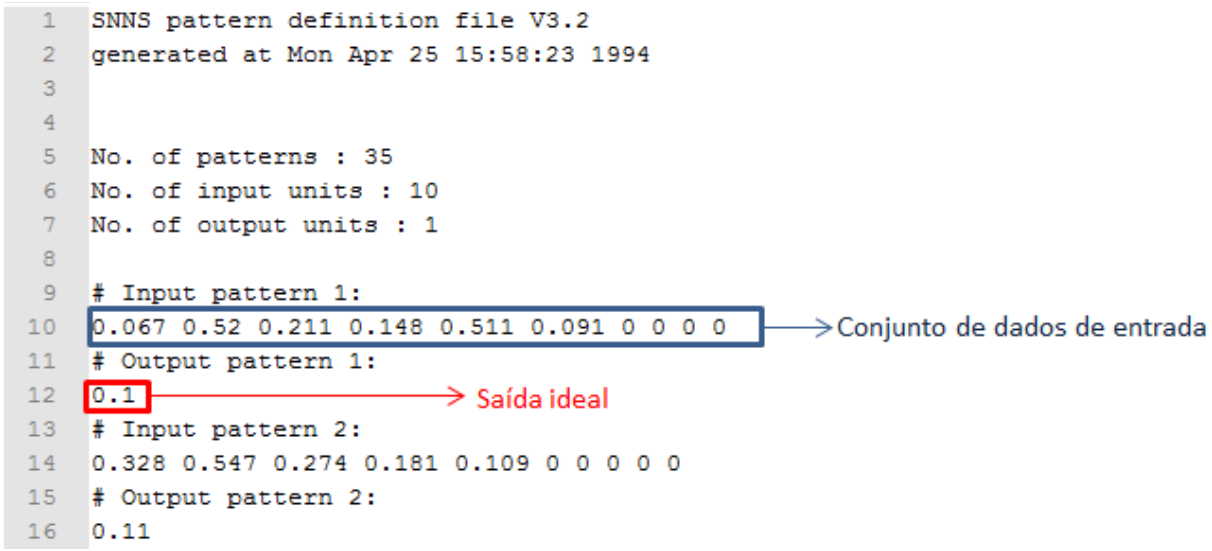

Figura 32 - Trecho do arquivo de padrões de treinamento.

Na figura 32 observa-se que a parte destacada como conjunto de dados de entrada é composto por dez elementos, representados por valores numéricos, os quais representam os sinais e sintomas. Estes elementos são as entradas da rede, compostos por dez, porque a rede possui dez neurônios em sua camada de entrada. Quando um elemento é representado por zero, significa a ausência de um sinal e sintoma. O *JavaNNS* necessita que todos os neurônios criados na rede recebam valores então, como forma de representar um campo inexistente, foi atribuído o valor de zero para sua representação, o menor valor que pode ser atribuído a um neurônio, pois foi definido o conjunto entre 0 e 1; isso ocorre diversas vezes no arquivo de padrão, pois muitas patologias apresentam menos de dez sinais e sintomas.

A utilização de dez neurônios na camada de entrada deve-se ao fato que, ao verificar todas as patologias utilizadas neste trabalho, foi identificado que dez é o número máximo de sinais e sintomas para uma patologia. Como a rede tem que receber todo o conjunto de dados, tanto para treinamento quanto validação, foi definida esta quantidade de neurônios na camada de entrada.

Ainda observando a figura 32 na parte destacada como saída ideal, esta é composta por apenas um elemento representado por valor numérico, que especifica uma patologia apontada pelos sinais e sintomas da camada de entrada. Desta forma, o arquivo de padrões é formado por elementos que representam os sinais e sintomas identificando a sua saída, que é a saída ideal. Saída ideal é a forma de expressar que, em um caso ideal quando inseridas as entradas ou sinais e sintomas idênticos a um padrão, a saída ou patologia deve ser a que estes sinais e sintomas se referem.

Assim, as redes neurais devem conseguir trabalhar com entradas desconhecidas. Por exemplo, a RNA deve conseguir, de acordo com o aprendizado adquirido pelo treinamento, uma aproximação com uma relação de sinais e sintomas que ainda não foi treinada, o que é chamado de generalização: quando uma rede neural consegue identificar uma patologia que se assemelha a entradas desconhecidas, mesmo não sendo a ideal.

A rede neural artificial não consegue trabalhar de forma ideal a todo o momento, o que pode acontecer é a diminuição do erro, que é a diferença entre a saída da rede comparada com a saída ideal. Quando isso ocorre é possível encontrar resultados próximos aos ideais e evitar grandes disparidades nos resultados. Na próxima seção será apresentada a criação e treinamento da rede neural, que utiliza o arquivo de padrões apresentados nesta seção.

4.4. Criação e treinamento da rede neural

Para a criação e treinamento foram utilizadas quatro redes neurais, com o objetivo de identificar uma estrutura de rede neural que apresente o melhor desempenho para a aplicação. Para tanto, seguiu-se as premissas apresentadas anteriormente, como o aprendizado supervisionado e respeitando o formato do arquivo de padrões e validação, além de seguir uma série de parâmetros para sua criação, apenas modificando a camada oculta da rede. Sendo assim, os parâmetros para criação das redes neurais artificiais (RNAs) são:

- Camada de Entrada ou *input*: são dez (10) neurônios ativados pela função de ativação *Act Identity* e utilizando a função de saída *Out Identity*.
- Camada Oculta ou *hidden*: neurônios ativados pela função de ativação *Act Logistic* e utilizando a função de saída *Out Identity*.
- Camada de Saída ou *output*: um (1) neurônio ativado pela função de ativação *Act Logistic* e utilizando a função de saída *Out Identity*.
- Tipo de conexões: *feed-forward*. Apresentado na seção 2.3.2.
- Inicialização: inicializada com valores entre 0 e 1, utilizando a função de inicialização *Random Weights*.
- Função de atualização: *topológical order*.
- Função de Aprendizado: *backpropagation*, com taxa de aprendizado (n) de 0.2 e erro acumulado (dmax) de 0.1.
- Ciclos de Treinamento: 5000 SSE.
- Quantidade de passos: 1.

- *Shuffle* ativado, mecanismo para tirar a ordem do arquivo de padrões e fazendo que os padrões sejão lidos aleatoriamente
- Arquivo de Padrões de Treinamento: training.pat com 45 padrões de entrada.
- Arquivo de Padrões de Validação: validation.pat com 12 padrões de entrada.

Todos os parâmetros apresentados anteriormente são idênticos para as quatro redes neurais artificiais (RNAs), porém diferenciando-se no aspecto da camada escondida, para comparar o resultado obtido entre ambas e determinar qual obtém melhor eficiência.

A alteração da camada escondida para análise das quatro redes neurais será o único parâmetro diferenciado entre as redes. A alteração dos outros parâmetros, como a camada de entrada da rede e saída necessitaria da alteração do arquivo de padrões, o que não se aplica ao objetivo desta comparação, que é a análise da alteração dos neurônios internos da rede encontrados na camada escondida.

4.4.1. RNA01

A criação da RNA01 segue os padrões apresentados na seção anterior. Diferencia-se por esta apresentar apenas um neurônio na camada oculta. Na figura 33 é observado o formato da rede neural treinada.

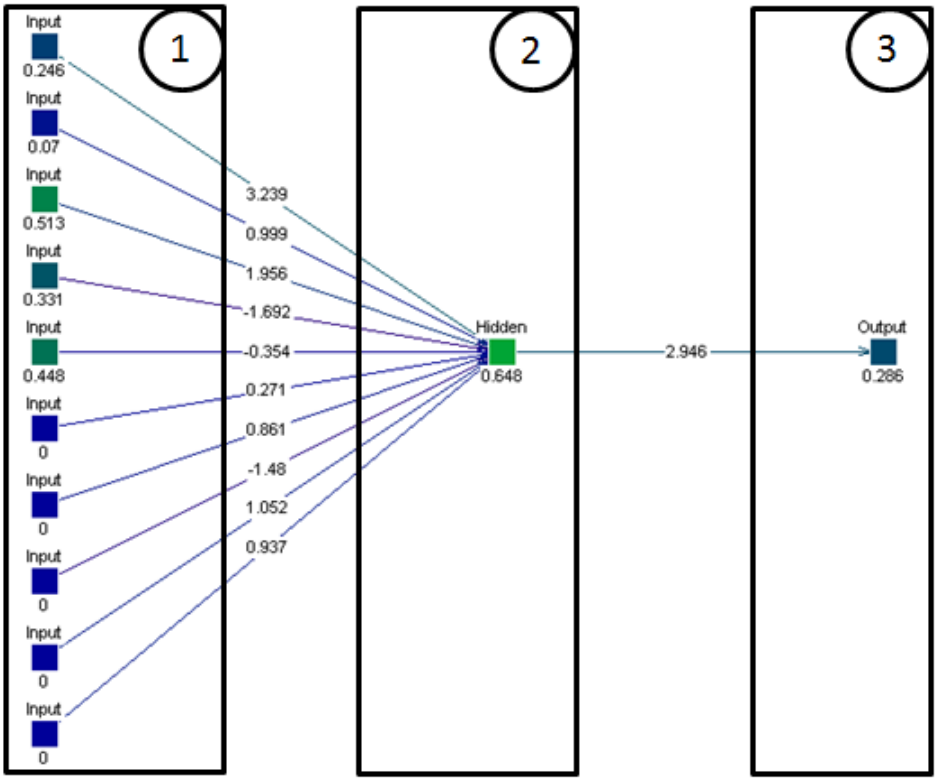

Figura 33 - Estrutura da RNA01.

Na figura 33 pode ser observada a estrutura da RNA01 dividida em três retângulos: no retângulo 1 são apresentadas as entradas da rede, formadas por dez neurônios descritos como *Input*; no retângulo 2 encontra-se a camada escondida formada por um neurônio descrito como *Hidden*; por fim, no retângulo 3 fica a saída da rede representado por um neurônio descrito como *Output*.

Para o treinamento da rede neural são utilizados os arquivos de padrões, o de treinamento e o de validação, que são utilizados várias vezes pela rede. Os ciclos de treinamentos são definidos por cada período que a rede utiliza o arquivo de padrões de treinamento para conseguir chegar a uma generalização dos padrões de validação. Em determinados momentos a rede vai conseguindo manter uma tendência em seu desvio, o que faz esta generalizar melhor os dados de entrada, que são dados que não são utilizados no treinamento, conseguindo uma aproximação para um resultado de acordo com as entradas fornecidas.

A convergência da rede é definida quando o desvio ou erro da curva de treinamento consegue se aproximar do desvio da curva de validação e ambas mantém uma tendência em seus índices de erro. Essa tendência do desvio muitas vezes se mantém e a rede consegue resultados próximos aos esperados. Quando a rede não consegue uma convergência ela apresenta altos índices de erros, acarretando em resultados com grandes disparidades.

Com o treinamento da RNA01 é possível obter o gráfico de erro que demonstra a relação entre erro da rede com a quantidade de treinamento realizado. O gráfico de erro da RNA01 pode ser observado na figura 34.

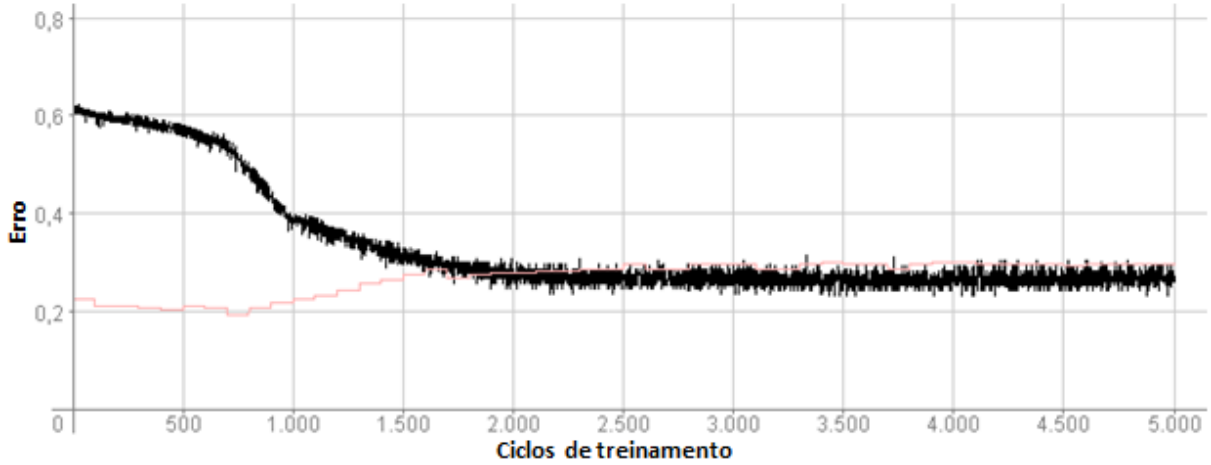

Figura 34 - Gráfico de erro da RNA01.

Na figura 34 pode ser observado gráfico de erro da RNA01. Partindo do ponto zero em sentido vertical são representados os valores de erro, e no sentido horizontal a quantidade de treinamento. Observa-se ainda que existem duas curvas, uma com cor preta partindo do ponto 0,6, que representa o treinamento da rede. Esta curva é cheia de altos e baixos, pois representa o erro de cada padrão utilizado no treinamento da rede. Existe também outra mais clara, partindo acima do ponto 0,2, que representa a validação da rede. A partir de 1500 ciclos de treinamento estas linhas começam a se convergir o que significa que a rede consegue, a partir do treinamento, generalizar a validação da rede. Quanto mais próximas as curvas, melhor é a generalização ou aproximação do resultado ideal apresentada pela rede.

Ainda na figura 34 pode ser observado que, mesmo com a convergência da rede, o erro se mantém entre 0,2 e 0,4 e permanece assim até o fim quando atinge 5000 ciclos de treinamento. Para um caso ideal, a rede neural teria que permanecer em 0 de erro em todos os ciclos de treinamento, porém também não haveria a necessidade do treinamento se a rede não apresentasse a variação de erro, pois ela já seria considerada ideal ou perfeita.

A cada interação de ciclos de treinamento a rede minimiza o seu desvio ou erro, porém ocorre que esta rede permanece uma tendência de desvio ao longo das outras interações, além de sofrer com alguns picos variando seu valor, o que torna necessária a realização de um cálculo de erro médio para melhor determinar sua eficiência. A figura 35 demonstra o arquivo de *log* da RNA01 que apresenta, através de valores, o erro apresentado pela rede, assim fica mais fácil perceber a mudança de valores a cada ciclo de treinamento.

```
opened at: Thu May 24 09:23:13 BRT 2012
Step 500 SSE:   0.5602929592132568      validation:     0.2028861790895462
Step 1000 SSE: 0.38095852732658386      validation:     0.21562537550926208
Step 1500 SSE: 0.29709672927856445      validation:     0.2653157413005829
Step 2000 SSE: 0.2870938181877136       validation:     0.277895450592041
Step 2500 SSE: 0.2655864357948303       validation:     0.28472793102264404
Step 3000 SSE: 0.2544044554233551       validation:     0.29650214314460754
Step 3500 SSE: 0.2822446823120117       validation:     0.2993599772453308
Step 4000 SSE: 0.2588500678539276       validation:     0.3005422055721283
Step 4500 SSE: 0.27257978916168213      validation:     0.2952972948551178
Step 5000 SSE: 0.2607976794242859       validation:     0.29699546098709106
```

Figura 35 - Arquivo de log da RNA01.

Na figura 35 é observado que o arquivo de *log* gerado pela RNA01 apresenta os erros de treinamento e validação a cada interação da rede. Percebe-se que os erros do treinamento iniciam bem mais altos do que os de validação, porém a cada interação os resultados vão diminuindo e, em determinadas interações, como a partir de 1500 ciclos, estes ficam bem

próximos, e logo se mantém em uma faixa. O arquivo de *log* traz com mais detalhes os valores de erro que são vistos também na figura 35. Na tabela 3 são apresentados os dados obtidos pela rede neural RNA01.

Tabela 3: Dados de treinamento da RNA01.

RNA	Erro Médio Treinamento	Erro Médio Validação	Maior Erro Treinamento	Maior Erro validação	Menor Erro treinamento	Menor Erro validação
RNA01	0,3115	0,2729	0,56	0,3	0,254	0,202

Na tabela 3 podem ser observadas algumas informações obtidas a partir dos resultados de treinamento da rede neural RNA01, que apresenta o seu erro médio do treinamento e validação, como também o maior e menor erro encontrado nas interações da rede. Com estas informações, pode ser feita uma comparação com as redes neurais RNA02, RNA03 e RNA04 para verificar a diferença de resultado que ambas podem apresentar.

4.4.2. RNA02

A RNA02 segue os padrões da RNA01, porém nesta rede neural a camada oculta é formada por três neurônios, não tendo sido modificadas as entradas e saídas da rede. A figura 36 ilustra a estrutura da RNA02 treinada.

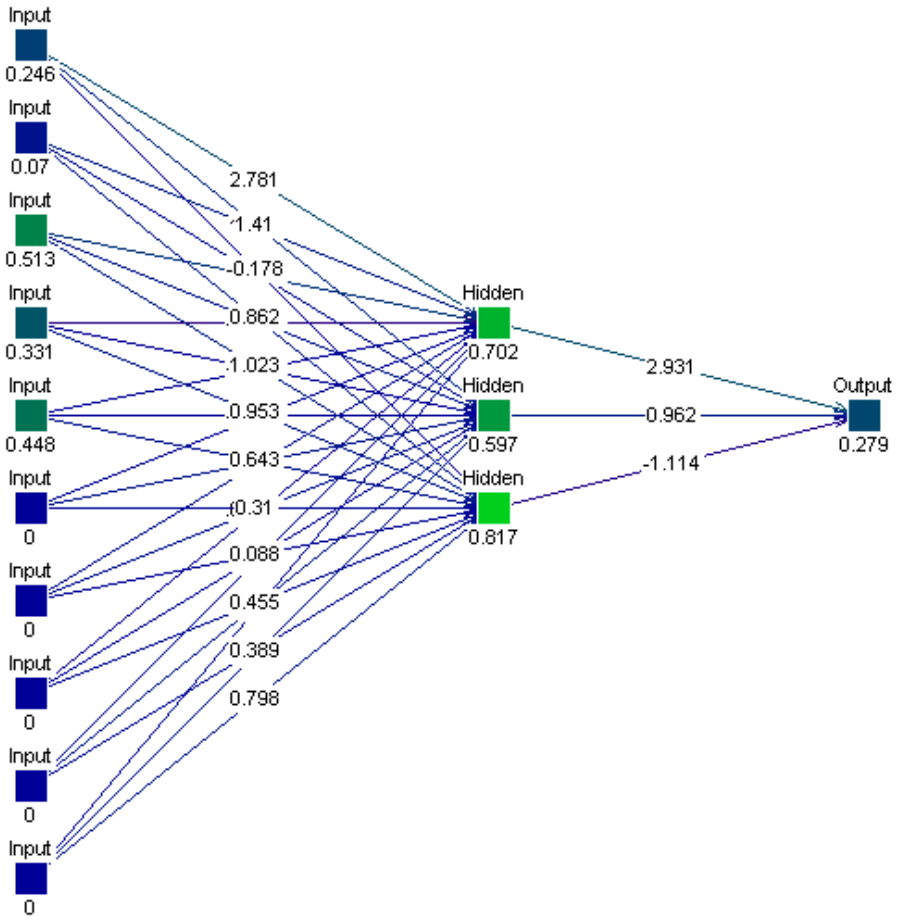

Figura 36 - Estrutura da RNA02.

Na figura 36 pode ser observado que a RNA02 não sofre grandes modificações em sua estrutura, pois a única modificação apresentada foi em sua camada oculta, que é formada por uma camada com três neurônios. Esta modificação pode alterar significativamente o desempenho da rede neural. O resultado desta operação é ilustrado na figura 37, que apresenta o gráfico de erro da RNA02.

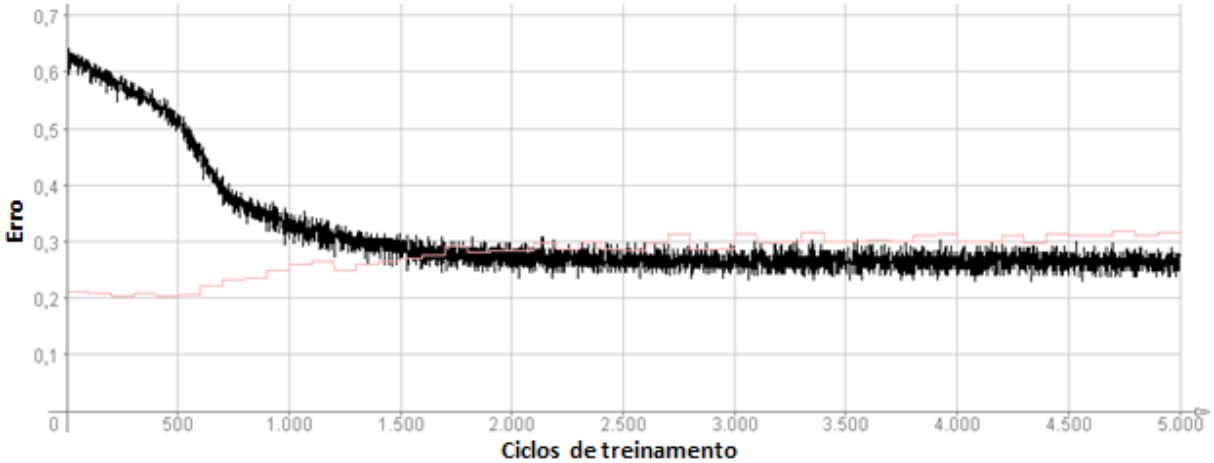

Figura 37 - Gráfico de erro da RNA02.

Na figura 37 pode ser observado o gráfico de erro da RNA02. Percebe-se que a RNA02 consegue convergir mais rapidamente que a RNA01, pois quando a rede atingiu 1000 ciclos de treinamento, as curvas começam a se unir, o que só ocorre a partir de 1500 ciclos na RNA01. Porém, a rede aparenta apresentar um erro superior do que a RNA01, o que pode significar um desempenho inferior que a RNA01, mesmo que aparentemente os resultados sejam parecidos. Pode ser observado na figura 38 o arquivo de *log* da RNA02 apresentando estes resultados.

```
opened at: Sat May 26 11:25:23 BRT 2012
Step 500 SSE:  0.40371644496917725      validation:     0.22669346630573273
Step 1000 SSE: 0.287449449300766        validation:     0.26691702008247375
Step 1500 SSE: 0.2766716778278351       validation:     0.29273027181625366
Step 2000 SSE: 0.2695235311985016       validation:     0.2866331934928894
Step 2500 SSE: 0.28149470686912537      validation:     0.3034001886844635
Step 3000 SSE: 0.2460712194442749       validation:     0.3010196387767792
Step 3500 SSE: 0.25520119071006775      validation:     0.301036536693573
Step 4000 SSE: 0.28515616059303284      validation:     0.2993920147418976
Step 4500 SSE: 0.2740563154220581       validation:     0.2996635437011719
Step 5000 SSE: 0.27142032980918884      validation:     0.307578444480896
```

Figura 38 - Arquivo de log da RNA02.

Na figura 38 observam-se os valores de erro apresentados pela RNA02 a cada interação. Se comparados com os valores apresentados pela RNA01, percebe-se que a RNA02 tem uma diminuição de erro inicial mais elevada que a rede neural RNA01, o que a torna mais contínua, diminuindo seu erro médio. Na tabela 4 são apresentadas as informações obtidas pelo treinamento da rede neural RNA02.

Tabela 4: Dados de treinamento da RNA02.

RNA	Erro Médio Treinamento	Erro Médio Validação	Maior Erro Treinamento	Maior Erro validação	Menor Erro treinamento	Menor Erro validação
RNA02	0,2847	0,288	0,403	0,307	0,246	0,226

Na tabela 4 pode ser observado que a RNA02 consegue atingir um valor de erro médio no treinamento melhor que a RNA01 e, em determinada interação, atinge o menor erro médio de treinamento se comparado com a RNA01. A presença de resultados melhores que a RNA01 pode ser destacada pela rápida convergência da rede, o que afeta os parâmetros de desvio da rede neural.

4.4.3. RNA03

A RNA03 segue os padrões da RNA01 e RNA02, porém é formada por duas camadas ocultas com três neurônios em cada camada. A figura 38 ilustra a estrutura da RNA03 treinada.

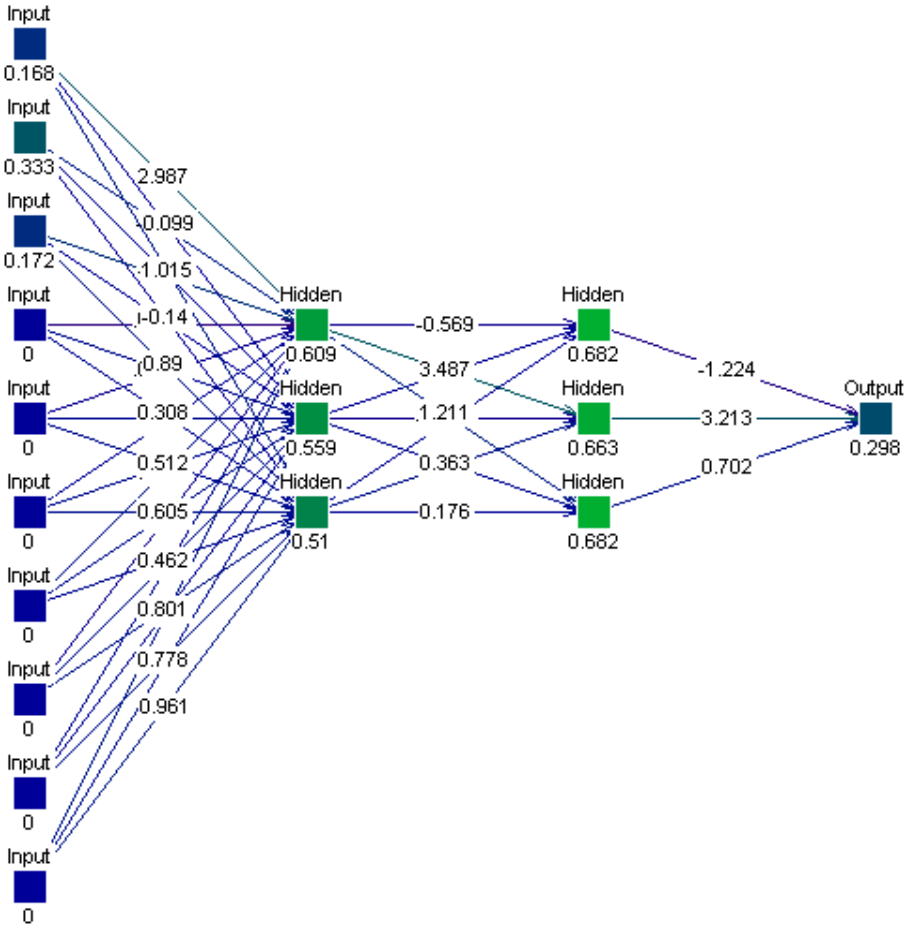

Figura 39 - Estrutura da RNA03.

Pode ser observado na figura 39 que a estrutura da RNA03 é diferente das RNA01 e RNA02, pois é formada por duas camadas ocultas com três neurônios em cada. Mesmo com esta modificação, como pode ser observada na figura 40 que ilustra o gráfico de erro da RNA03, a rede necessitou de mais ciclos de treinamento para conseguir convergir, se comparada com as RNA01 e RNA02.

Figura 40 - Gráfico de erro da RNA03.

Na figura 40 pode ser observado o gráfico de erro da RNA03 que apresenta uma convergência da rede mais atrasada em relação as RNA01 e RNA02, pois depois de 3000 ciclos de treinamento a rede consegue se convergir. A figura 40 ilustra o arquivo de *log* da RNA03 com os resultados dos treinamentos.

```
ppened at: Sat May 26 14:14:26 BRT 2012
Step 500 SSE:  0.6049696803092957      validation:    0.21101856231689453
Step 1000 SSE: 0.606363832950592       validation:    0.21168574690818787
Step 1500 SSE: 0.5970543026924133      validation:    0.20921453833580017
Step 2000 SSE: 0.5741804838180542      validation:    0.2127637416124344
Step 2500 SSE: 0.40962857007980347     validation:    0.2081783264875412
Step 3000 SSE: 0.3122413456439972      validation:    0.2606765627861023
Step 3500 SSE: 0.27262312173843384     validation:    0.27653858065605164
Step 4000 SSE: 0.2783638536930084      validation:    0.2901231646537781
Step 4500 SSE: 0.268667608499527       validation:    0.2919658124446869
Step 5000 SSE: 0.2771437466144562      validation:    0.29230383038520813
```

Figura 41 - Arquivo de log da RNA03.

Na figura 41 observa-se o arquivo de *log* da RNA03, apresenta inicialmente um desempenho inferior as RNA01 e RNA02, o que permite inferir que o aumento das camadas ocultas, nesta configuração, não melhora o desempenho da rede, além de exigir um tempo maior para a convergência da rede. Na tabela 5 podem ser observadas informações do treinamento e validação da RNA03

Tabela 5: Dados de treinamento da RNA03.

RNA	Erro Médio Treinamento	Erro Médio Validação	Maior Erro Treinamento	Maior Erro validação	Menor Erro treinamento	Menor Erro validação
RNA03	0,4197	0,246	0,606	0,292	0,268	0,208

Na tabela 5, pode ser observado que o erro médio no treinamento da RNA03 teve um aumento significativo, além de elevar os outros parâmetros analisados, o que demonstra que o aumento no número de camadas ocultas tornou a rede mais lenta, com índices de erro mais elevados, por analisar mais a relação de entradas.

4.4.4. RNA04

A RNA04 segue os padrões da RNA01, RNA02 e RNA03 descritos anteriormente, porém é formada por duas camadas ocultas com seis neurônios em cada camada. A figura 41 ilustra a estrutura da RNA03 treinada.

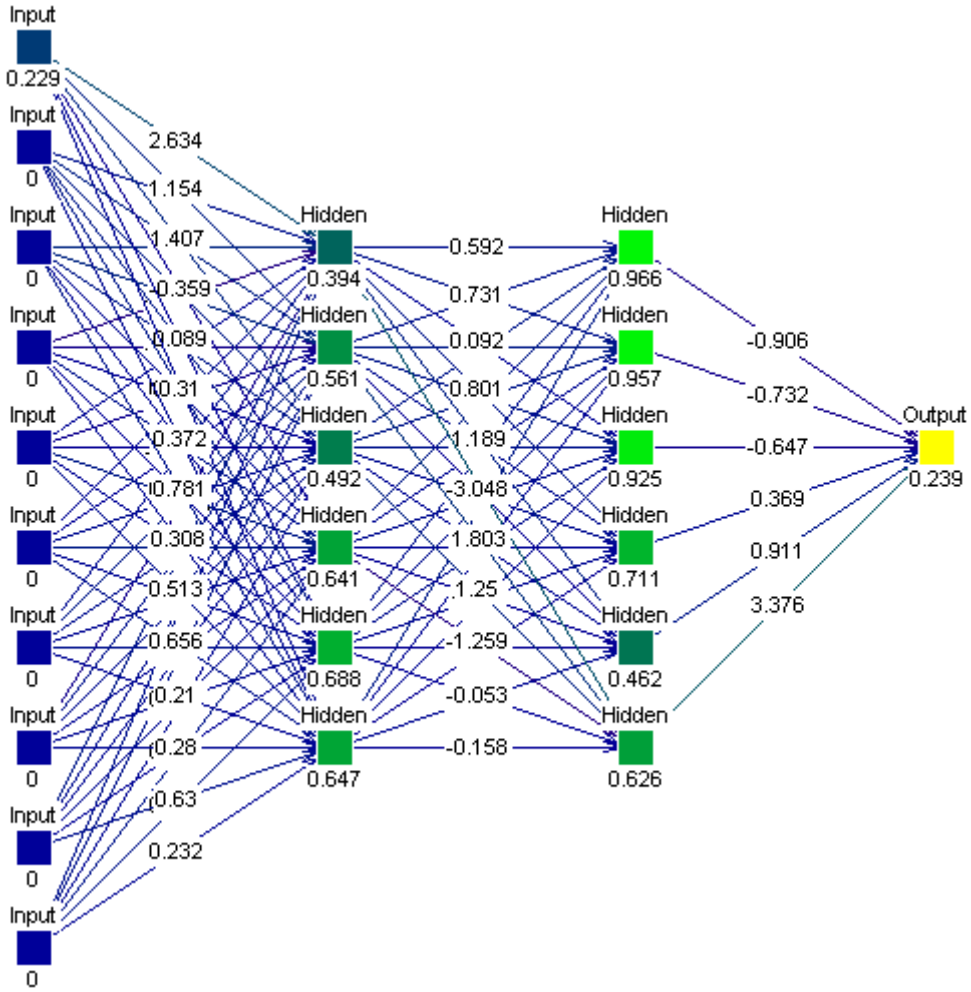

Figura 42 - Estrutura da RNA04.

Na figura 42 pode ser observada a estrutura da RNA04 formada por duas camadas ocultas e seis neurônios em cada camada. Mesmo com essa mudança a rede não consegue

obter melhor desempenho que as demais, como se pode perceber analisando seu gráfico de erro observado na figura 43.

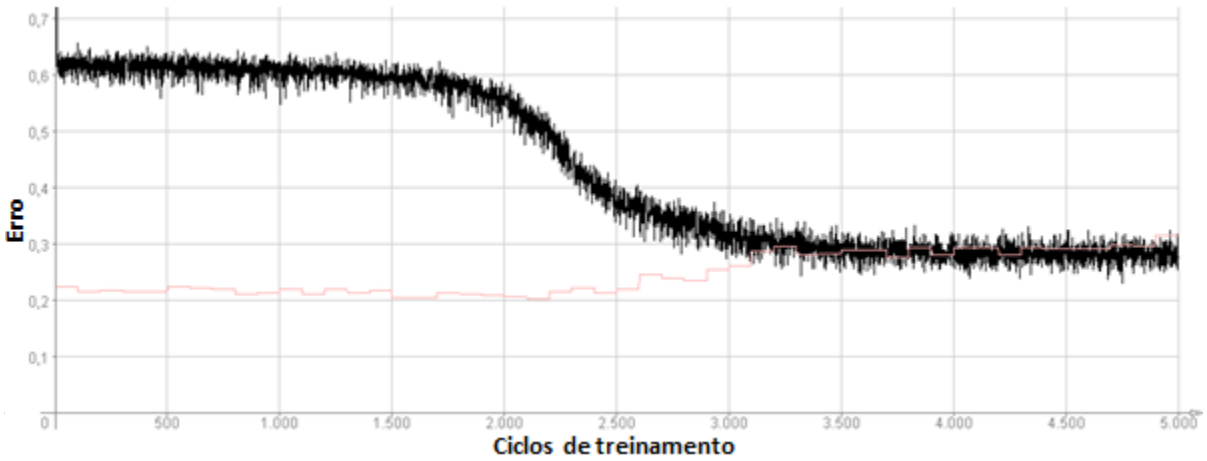

Figura 43 - Gráfico de erro da RNA04.

Na figura 43 pode ser observado o gráfico de erro da RNA04. Assemelhando-se a RNA03, seu tempo de convergência foi alto e em relação às RNA01, RNA02 e RNA03 teve desempenho menor, comprovado no seu arquivo de *log*, que pode ser observado na figura 44.

```
opened at: Sat May 26 14:20:07 BRT 2012
Step 500 SSE:   0.6198928952217102      validation:     0.21561844646930695
Step 1000 SSE: 0.6179274916648865       validation:     0.21341747045516968
Step 1500 SSE: 0.5925678610801697       validation:     0.21620361506938934
Step 2000 SSE: 0.5666035413742065       validation:     0.20717759430408478
Step 2500 SSE: 0.3708341121673584       validation:     0.21227972209453583
Step 3000 SSE: 0.29860273003578186      validation:     0.25485140085220337
Step 3500 SSE: 0.29488620162010193      validation:     0.2823601961135864
Step 4000 SSE: 0.27267369627952576      validation:     0.2807992696762085
Step 4500 SSE: 0.28108957409858704      validation:     0.29139643907546997
Step 5000 SSE: 0.2754409909248352       validation:     0.3150559663772583
```

Figura 44 - Arquivo de log da RNA04.

Na figura 44, observa-se o arquivo de *log* da RNA04 apresenta uma generalização mais baixa que as demais redes, o que interfere no seu desempenho e faz com que a rede necessite de mais ciclos de treinamento para melhorar o seu desempenho. Na Tabela 6 são apresentados os índices apresentados pela RNA04.

Tabela 6: Dados de treinamento da RNA04.

RNA	Erro Médio Treinamento	Erro Médio Validação	Maior Erro Treinamento	Maior Erro validação	Menor Erro treinamento	Menor Erro validação
RNA04	0,4184	0,2485	0,619	0,315	0,272	0,207

Na tabela 6, pode ser observado que os índices de erros mais elevados continuam, se comparados com a RNA03, além de apresentar o maior erro registrado em uma interação no treinamento e validação, comparando com RNA01, RNA02, RNA03. Na próxima seção é apresentado um comparativo das redes neurais apresentadas anteriormente, com as informações retidas dos parâmetros de treinamento e validação como forma de determinar qual rede pode ser considerada a que possui melhor desempenho.

4.5. Analise dos Resultados das Redes Neurais.

Ao analisar os resultados das Redes Neurais Artificiais (RNAs) e comparando uma com as outras, pode-se perceber que as redes mais complexas tiveram um desempenho menor que as outras, além de um processamento maior, ocorrido pela camada extra. Pode ser observada a comparação das quatro redes neurais na figura 45.

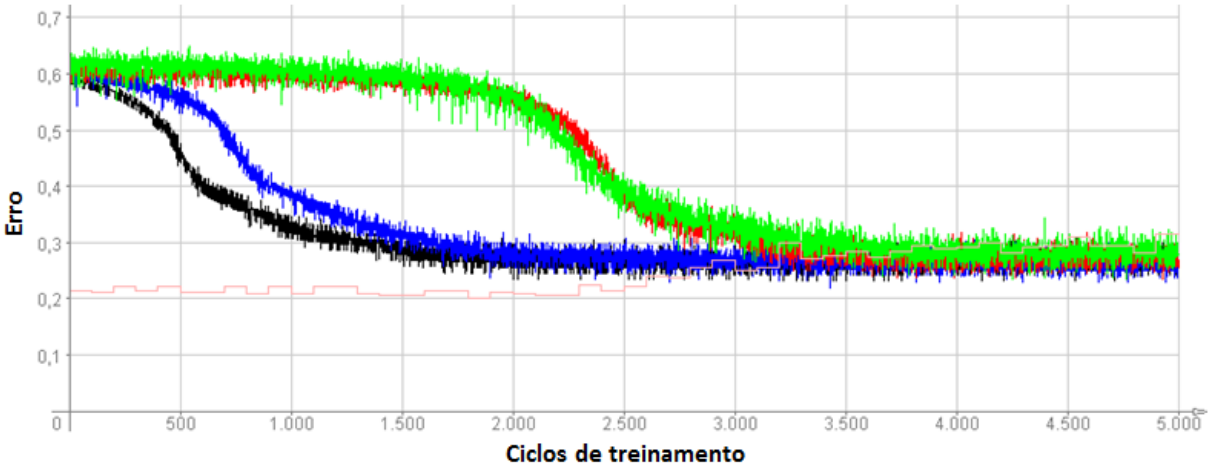

Figura 45 – Gráficos de comparação das 4 RNAs criadas.

Na figura 45, pode ser observado o gráfico de erro das quatro redes neurais criadas neste trabalho. As redes neurais RNA01, RNA02, RNA03 e RNA04 são representadas respectivamente pelas cores azul, preto, vermelho e verde. Além disto, existe a linha de validação representada por uma linha mais fina partindo um pouco acima do ponto 0,2. Observando a figura 45, percebe-se que a RNA02 converge mais rápido que as demais redes, acompanhada pela RNA01. As RNA03 e RNA04 utilizam mais ciclos de treinamento para conseguir convergir, por isso demoram mais a atingir níveis menores de erro.

Uma rede que demora a convergir, como as RNA03 e RNA04, apresenta um valor de erro alto por várias interações, o que afeta seu erro médio, além de necessitar de ciclos de

treinamento mais longos para conseguir convergir. Se para as redes neurais fossem atribuídos apenas 2000 ciclos de treinamento, as RNA03 e RNA04 não conseguiriam convergir adequadamente, além do que possuiriam um alto índice de erro afetando a generalização dos dados que fossem testados na rede. O que ocorre inversamente nas RNA01 e RNA02, que conseguiram convergir e atingir índices menores de erro com 2000 ciclos de treinamento. Pode-se verificar na figura 45 que as RNAs mantêm-se em uma faixa de erro até completarem os 5000 ciclos de treinamento, mesmo ocorrendo pequenas variações. Na tabela 7 pode ser observada a comparação das informações apresentadas pelas redes neurais.

Tabela 7: Informações das 4 redes neurais treinadas.

RNA	Erro Médio Treinamento	Erro Médio Validação	Maior Erro Treinamento	Maior Erro Validação	Menor Erro Treinamento	Menor Erro Validação
RNA01	0,3115	0,2729	0,56	0,3	0,254	**0,202**
RNA02	**0,2847**	0,288	0,403	0,307	**0,246**	0,226
RNA03	0,4197	**0,246**	0,606	0,292	0,268	0,208
RNA04	0,4184	0,2485	**0,619**	**0,315**	0,272	0,207

Na tabela 7 pode ser observado que a RNA02 apresentou o menor erro médio, seguido das redes RNA01, RNA04 e RNA03, além de apresentar o menor erro no treinamento em suas interações. Pelas informações apresentadas, pode-se determinar que a RNA02 possui o melhor desempenho comparado às RNA01, RNA03 e RNA04, pois apresenta o menor erro na maioria dos parâmetros.

Foi feita a transformação das redes neurais RNA01, RNA02, RNA03, RNA04 em código na linguagem de programação C, que se encontra nos Anexos A, B, C e D respectivamente. A transformação destas redes em código tem como objetivo possibilitar a sua utilização em trabalhos futuros que necessitem utilizar estes códigos em suas implementações, como também analisar como é a sequência lógica de execução utilizando redes neurais para o domínio deste trabalho.

Na próxima seção será apresentada a validação do trabalho pelo especialista do domínio, em que, com a utilização de alguns testes, foi verificada a eficiência da aplicação de redes neurais artificiais para o diagnóstico de patologias traumato-ortopédicas dos membros inferiores. Para a validação foi utilizada a RNA02 que obteve melhor desempenho na comparação com as demais RNAs.

4.6. Validação com o especialista do domínio

Para a validação junto ao especialista do domínio, foram criados testes em que foram inseridos conjuntos de sinais e sintomas incompletos na rede neural para verificar as aproximações da rede e determinar a patologia indicada. Isto foi realizado com a intenção de que a rede apresentasse alguns resultados para serem analisados pelo especialista, permitindo que o mesmo apontasse se a rede apresenta alguns resultados que podem ter relação com a patologia esperada.

Para uma melhor avaliação do especialista, foram transformados os resultados da rede de acordo com sua representação real, ressaltando que a rede neural utiliza-se de dados numéricos para representar os sinais e sintomas e as suas respectivas patologias. Isto dificulta a avaliação por parte do especialista por necessitar de utilizar material de apoio para identificar os significados numéricos da rede. Assim, como forma de facilitar sua avaliação, foram realizados estes procedimentos.

A Rede Neural utilizada para estes procedimentos foi a RNA02, por apresentar melhor desempenho, como observado na seção anterior. Primeiramente, foi escolhida uma sequência qualquer de sinal e sintomas e a sua respectiva patologia. Em um dos testes foi utilizada a Doença de Legg-Calvé-Perthes, sendo retirado um de seus sintomas, a hipersensibilidade à palpação. Para realizar este teste deve ser feita a alteração no arquivo de validação onde as sequências que representam os sinais e sintomas hipersensibilidade à palpação são retirados e a rede é executada. Na tabela 8 podem ser observados os parâmetros completos e a modificação realizada.

Tabela 8: Parâmetros utilizados no teste da rede.

Sequência completa	0.373 0.328 0.088 0.388 0.277 0.115 0 0 0 0
Sequência incompleta	0.373 0 0.088 0.388 0.277 0.115 0 0 0 0
Saída esperada	0.16
Saída da rede	0.20405

Na tabela 8 pode ser observado que a sequência que determina a saída esperada da rede neural foi alterada, utilizando uma entrada incompleta, assim a rede busca alguma patologia que, a partir do seu treinamento, considera a correta. A rede neural consegue determinar a saída através da aproximação e, de acordo com as modificações da entrada, a rede neural apresenta uma saída dentro de seu desvio, o que pode significar em uma saída válida, pois a rede consegue uma aproximação do resultado esperado. Na tabela 9 pode ser observada a saída esperada da rede com seus respectivos sinais e sintomas.

Tabela 9: Patologia Legg-Calvé-Perthes com a representação de seus sinais e sintomas.

Doença de Legg-Calvé-Perthes	Inicio gradual da dor no quadril, na região inguinal, na coxa e no joelho, especialmente com a sustentação do peso corporal	0,373
	Hipersensibilidade à palpação	0,328
	Abdução, rotação interna e externa reduzidas	0,088
	Marcha de Trendelenburg	0,388
	Espasmos musculares	0,277
	Atividades funcionais reduzidas	0,115

Na tabela 9, observa-se a Doença de Legg-Calvé-Perthes e seus respectivos sinais e sintomas com sua representação na RNA. Percebe-se que, para retirar determinado sinal e sintoma do arquivo de padrões, basta acrescentar o valor de 0, que representa a ausência de um sinal e sintoma na rede neural. Assim, a RNA busca através de seu aprendizado um resultado aproximado, sendo que neste caso a patologia resultante dos padrões inseridos foi a Entorse/laceração do ligamento colateral medial. Na tabela 10 pode ser observada a saída apresentada pela rede.

Tabela 10: Patologia Entorse/laceração do ligamento colateral medial com seus respectivos sinais e sintomas.

Entorse/laceração do ligamento colateral medial	Dor
	Flexão e extensão do joelho limitadas
	Tumefação
	Hipersensibilidade à palpação no local da entorse ou laceração
	Padrão da marcha alterado
	Pode desenvolver-se fraqueza na altura da coxa

Na tabela 10 observa-se a patologia que a RNA apresenta, sendo que alguns sinais e sintomas são parecidos, mas que o especialista ressalta que mesmo parecidas, a região do corpo humano onde se encontram é diferente. Porém como a rede só utilizou padrões de treinamento e ainda não trabalhou com dados incompletos, existe procedência da RNA apresentar tal resultado, pois a mesma permaneceu em seu desvio.

Ao avaliar o resultado da rede o especialista descreve que o resultado não é o ideal, mas existe lógica da rede neural apresentar tal patologia, por a mesma apresentar sinais e sintomas que parecem semelhantes. Além deste teste foram feitos outros que seguem este mesmo exemplo, em que o especialista mantém sua conclusão.

Ao analisar as quatro redes neurais RNA01, RNA02, RNA03 e RNA04, a rede neural RNA02 apresentou a menor média de erro e conseguiu, com poucos ciclos de treinamento, a convergência, chegando a resultados mais aproximados, pois com o índice de desvio menor a rede consegue chegar a um resultado esperado, com maior frequência. Isto auxilia os especialistas da área a determinar o diagnóstico das patologias traumato-ortopédicas dos membros inferiores, já que a rede consegue opinar em resultados com desvio considerável.

O especialista do domínio apontou que o resultado inicial que as redes neurais apresentam possuem procedência pelos sinais e sintomas informados, mesmo que a rede apresente algumas patologias que muitas vezes serão desconsideradas pelos profissionais.

Em determinados casos uma patologia apontada deve ser considerada, mesmo que a rede apresente algumas patologias que muitas vezes não serão consideradas, pois às vezes existe a necessidade da verificação através de exames para eliminar estas patologias do diagnóstico. Como rede neural evolui de acordo com os treinamentos, o uso contínuo faz com que a rede apresente resultados cada vez mais próximo aos esperados.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou os conceitos e características da Inteligência Artificial (AI), como uma de suas técnicas, as Redes Neurais Artificiais (RNAs), além de apresentar a descrição das patologias traumato-ortopédicas dos membros inferiores do corpo humano, que são o domínio de aplicação, como também a ferramenta utilizada no processo de criação, treinamento e validação das RNAs, o JavaNNS. O estudo de todos estes conceitos é imprescindível para a compreensão da criação e desenvolvimento deste trabalho.

A RNA é uma técnica que apresenta um grande potencial em apresentar resultados satisfatórios, isso se deve ao fato de utilizar uma similaridade com o cérebro humano, tornando a RNA capaz de aprender e propor soluções para situações desconhecidas.

Aplicando as técnicas mencionadas anteriormente para o diagnóstico de patologias traumato-ortopédicas dos membros inferiores, o trabalho apresenta a possibilidade de, por meio dos sinais e sintomas, conseguir chegar às patologias, através de aproximação. Utilizando o JavaNNS como ferramenta para desenvolver este sistema, foi possível criar redes neurais que conseguem dentro de seu desvio padrão, indicar patologias mesmo que não esperadas, mas próximas as saídas ideais.

Para poder validar o resultado obtido através da RNA, foi necessária a presença do especialista, que é a pessoa responsável por analisar e avaliar, dentro de seu domínio, se o que se espera do sistema foi alcançado, e determinar correções que possam surgir, pela execução destes procedimentos.

Ao analisar as redes neurais após o treinamento, verificou-se que a rede neural RNA02 obteve o melhor desempenho comparado com as demais redes neurais, e utilizando RNA02 para teste analisou que a rede fornece saídas com boa aproximação às saídas esperadas. Sabendo que as redes neurais utilizadas neste trabalho utilizam da aproximação para determinar um resultado, além de verificar que os resultados mesmo que às vezes não condizentes com a patologia esperada, tem relação através de seus sinais e sintomas.

Como sugestão para trabalhos futuros, pode ser realizada a implementação do sistema, com interface para que os usuários tenham interação com as funções da aplicação, formulários para manipulação dos dados inserindo os sinais e sintomas para a rede, relatórios com os resultados fornecidos pela rede neural e auxiliar na visualização dos resultados, no qual

possa fornecer um ambiente em que usuários possam ter acesso as informações da rede. Em anexo, encontra-se a representação em linguagem C das quatro redes neurais utilizadas neste trabalho, o que pode facilitar nesta implementação.

Outros trabalhos podem utilizar formatos diferentes de redes neurais para comparação de seus resultados, com o objetivo de identificar estruturas de redes neurais com melhor eficiência para este domínio. A comparação com outras técnicas da Inteligência Artificial (IA) também pode ser sugerido, pois a inteligência artificial possui varias técnicas, o qual podem fornecer outros resultados, podendo apresentar melhor desempenho no diagnóstico destas patologias.

É importante ressaltar que o objetivo da rede neural deste trabalho é auxiliar os profissionais da área em seu diagnóstico, portanto a rede neural não deve ser utilizada como ferramenta essencial para o diagnóstico, pois somente com o conhecimento e experiência do profissional habilitado é possível chegar a real patologia. Entretanto, pode-se concluir que a rede neural, como proposta, pode ser uma ferramenta de grande apoio ao trabalho deste profissional.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ARONE, Evanisa Maria. **Enfermagem médico-cirurgica aplicada ao sistema nervoso. 7ª edição revisada e ampliada**, São Paulo: Editora Senac, 2005, 135p.

BALLONE, GJ . **Neurônios e Neurotransmissores**. Disponível em: <http:// www.psiqweb.med.br/site/?area=NO/LerNoticia&idNoticia=290>. Acesso em: 26 jul 2012.

BARCA, Maria Carolina Stockler; SILVEIRA, Tiago Redondo de Siqueira; MAGINI, Marcio. Treinamento de redes neurais artificiais: o algoritmo Backpropagation. In:
IX Encontro Latino Americano de Iniciação Científica, V Encontro Latino Americano de Pós-Graduação – Universidade do Vale do Paraíba, **Anais**. Jacareí, 2005. Disponível em: <http:// biblioteca.univap.br/dados/INIC/cd/inic/IC1%20anais/IC1-17.pdf>. Acesso em: 8 nov. 2011.

BRAGA, Antonio; LUDEMIR, Teresa; CARVALHO, André. **Redes Neurais Artificiais**: Teoria e Aplicações. Rio de Janeiro: LTC, 2000. 262 p.

BRASIL. resolução nº. 260, de 11 de fevereiro de 2004, Conselho Federal de Fisioterapia e Terapia Ocupacional. **COFFITO**, Seção 1, p. 66-67.

BOCANEGRA, Charlie W. R. **Procedimentos para tornar mais efetivo o uso das redes neurais artificiais em planejamento de transportes**. Dissertação (Mestrado). Escola de Engenharia de São Carlos - Universidade de São Paulo. São Carlos, 2002. 108 p.

BORBA, Anderson**. Ciências Morfológicas.** Disponível em: < http:// cienciasmorfológicas.webnode.pt/introdu%C3%A7%C3%A3o%20a%20anatomia/ divis%C3%A3o%20do%20corpo%20humano/>. Acesso em: 8 nov. 2011.

COFFITO. **Definição de Fisioterapia**. Disponível em: <http://coffito.org.br/conteudo/ con_view.asp?secao=27>. Acesso em: 29 mai. 2012.

COPPIN, Bem. **Inteligência Artificial**. Rio de Janeiro: LTC, 2010, 636 p.

FARACO, Rosemary Antonia Lopes; COSTA JR, Pyramo Pires; CRUZ, Frederico Rodrigues Borges. Minimização do erro no algoritmo Back-Propagation aplicado ao problema de manutenção de motores. **Pesquisa Operacional,** Belo Horizonte-MG, v.18, n. 1, jun de 1998.

GANN, Nancy. **Ortopedia**: guia de consulta rápida para Fisioterapia: distúrbios, testes e estratégias de reabilitação. Rio de Janeiro: Guanabara Koogan, 2005. 192 p.

HAYKIN, Simon. **Redes Neurais:** princípios e prática. 2 ed. Porto Alegre, RS: Bookman, 2001, 900 p.

HAYKIN, Simon. **Redes Neurais:** princípios e práticas. 2. ed. Porto Alegre, RS: Bookman, 1999, 903 p.

LUGER, George F. **Inteligência Artificial**: estruturas e estratégias para a solução de problemas complexos. 4. ed. Porto Alegre, RS: Bookmann, 2004, 774 p.

MANCUSO; Juliana Prestes. Fisioterapia Ortopédica: historia e definição. Disponível em: < http://www.fisioterapeutasplugadas.com.br/fisioortop.asp>. Acesso em: 8 nov. 2011.

MICHAELIS. Moderno Dicionário da Língua Portuguesa. Disponível em: <http:// michaelis.uol.com.br/moderno/ingles/index.php>. Acesso em: 9 nov. 2011.

NASCIMENTO JUNIOR, Cairo Lúcio. **Inteligência Artificial em controle e automação**. 1. ed. São Paulo, SP: Edgard Blücher, 2000, 218 p.

NETO, Sigismundo; NAGANO, Marcelo Seido; MORAES, Marcelo Botelho da Costa. Utilização de redes neurais artificiais para avaliação socioeconômica: uma aplicação em cooperativas. **RAUSP: Revista de Administração**, São Paulo-SP, v.41, n.1, p.59-68, jan./ fev./mar. 2006.

OLIVEIRA, Ivo Sócrates Moraes de. **Utilização de Redes Neurais Artificiais para a análise de seqüências de proteínas homólogas**. Trabalho de Conclusão de Curso. Centro Universitário Luterano de Palmas – Palmas, TO, 2007.

REZENDE, Solange. **Sistemas Inteligentes**: fundamentos e aplicações. 1Barueri, SP: Manole, 2005. 523 p.

RICH, Elaine; KNIGHT, Kevin. **Inteligência Artificial**. 2. ed. São Paulo, SP: Makron Books, 1993, 722 p.

ROSA, Thatiane Oliveira. **Aplicação de algoritmos genéticos para comparação de seqüências biológicas**. Trabalho de Conclusão de Curso. Centro Universitário Luterano de Palmas – Palmas, TO, 2007.

RUSSELL, Stuart; NORVING, Peter. **Inteligência Artificial**. 2. ed. Rio de Janeiro, RJ: Elsevier, 2004. 1021 p.

WOLFOVITCH, Moysés, et al. **A história da ortopedia no estado da Bahia**. Universidade Federal da Bahia – Salvador, 2007. Disponível em: <http://www.gmbahia.ufba.br/ojs/index.php/gmbahia/article/viewFile/105/98>. Acesso em 30/06/2011.

SOUZA FILHO, Marlos Roberto de. **Rede Neural Artificial para o deslocamento de um Robô Autônomo**. 2009. 210 f. Trabalho de Conclusão de Curso (Bacharel) – Curso de Ciência da Computação, Univali, Itajaí, 2009.

KARRER, Daniel; CAMEIRA, Renato Florido; VASQUES, André Strauss; BENZECRY, Marcos de Almeida. **Redes neurais artificiais: conceitos e aplicações**. Profundão – IX Encontro de Engenharia de Produção da UFRJ. 2005. Artigo. 12p.

MATRIAS, Ivo Mário, **Aplicação de redes neurais artificiais na análise de dados de molhamento foliar por orvalho**. Tese (Doutorado) Universidade Estadual Paulista, Faculdade de Ciências Agronômicas, Botucatu, 2006, 120 f.

GONZÁLEZ, Marco Aurélio Stumpf. **Aplicação de técnicas de descobrimento de conhecimento em bases de dados e de inteligência artificial em avaliação de imóveis**. Tese (Doutorado) PPGEC/UFRGS, 2002, 300 p. Porto Alegre.

ROGAL JÚNIOR, Sérgio Renato. **Detecção e classificação de arritmias cardíacas utilizando redes neurais artificiais auto-organizáveis**. Dissertação Mestrado, PUC-PR, Curitiba, 2008, 89p.

SEVERO, Diogo da Silva. **Otimização Global em Redes Neurais Artificiais**. Trabalho de Conclusão de Curso. UFPE – Recife, PE, 2010. Disponível em: < http://www.cin.ufpe.br/~tg/2010-1/dss2.pdf>. Acesso em 28 jun. 2012.

GUIMARÃES, Alaine M.; MATHIAS, Ivo M.; DIAS, Ariangelo H.; FERRARI Jones W.; CAZELATTO JUNIOR, Carlos R. De O. **Módulo de validação cruzada para treinamento de redes neurais artificiais com algoritmos Backpropagation e Resilient Propagation**. Publ. UEPG Ci. Exatas Terra, Ci. Agr. Eng., Ponta Grossa, 14 (1): 17-24, abr. 2008

YONENAGA, William Hajime; FIGUEIREDO, Reginaldo Santana. Previsão do preço da soja utilizando redes neurais. Universidade Federal de São Carlos.

_____. **Entenda o que acontece no seu cérebro**. Disponível em: < http://nah220kw.blogspot.com.br/2011/07/entenda-o-que-acontece-no-seu-cerebro.html.>. Acesso em: 26 jul 2012.

APÊNDICES

APÊNDICE A – Relação de patologias com respectivos sinais e sintomas.

Local: Quadril e coxa	
Patologias	**Sinais e Sintomas**
Doença articular degenerativa	A dor fica localizada ao redor da região inguinal e raramente irradia-se para o joelho
	Rigidez matinal <30 min
	Dor em queimação, em especial à noite e à atividade
	Desvios da marcha
	Retração capsular
	ADM e função limitadas
Distensões e lacerações musculares	Hipersensibilidade à palpação
	Tumefação
	Equimose
	Dor ao alongamento passivo e a ADM
	Anormalidades da textura tecidual
Bursite trocantérica	A dor tem início insidioso, localiza-se na parte lateral do quadril, e pode irradiar-se para o dermátomo L5, especialmente ao subirem escadas, à posição de decúbito lateral ou ao se sair de um carro
	A ADM costuma ser completa, porém a adução pode ser limitada
Pontada no quadril	Dor localizada na crista ilíaca
Meralgiaparestésica	A dor é de inicio gradual ou súbito e acomete a parte anterolateral da coxa
	A dor é em queimação, sendo agravada por adução e flexão forçada do quadril
	Hipoestesia e dormência ocasional
	A ADM em geral é plena, exceto para retração, e pode ser dolorosa no final da amplitude
	Hipersensibilidade à palpação
Miositeossificante	Dor local na área da contusão
	A ADM do joelho limitada, especialmente em flexão
	Podem ser palpadas anormalidades na textura tecidual
Doença de Legg-Calvé-Perthes	Inicio gradual da dor no quadril, na região inguinal, na coxa e no joelho, especialmente com a sustentação do peso corporal
	Hipersensibilidade à palpação
	Abdução, rotação interna e externa reduzidas
	Marcha de Trendelenburg
	Espasmos musculares
	Atividades funcionais reduzidas
Deslocamento da epífese da cabeça do fêmur	Dor leve à palpação no quadril, na perna, na coxa e no joelho, nos casos crônicos
	Discrepâncias no comprimento dos MMII (DCP)
	Atrofia do quadrípces
	ADM reduzida, especialmente em RI, abdução e flexão

	Marcha de Trendelenburg;
Luxação congênita do quadril	Abdução reduzida
	Perna mantida em flexão com abdução
Artroplastia total do quadril	Em geral muito dolorosa após a cirurgia
	Mobilidade e força limitadas
Local: Joelho	
Patologias	**Sinais e Sintomas**
Entorse/laceração do ligamento colateral medial	Dor
	Flexão e extensão do joelho limitadas
	Tumefação
	Hipersensibilidade à palpação no local da entorse ou laceração
	Padrão da marcha alterado
	Pode desenvolver-se fraqueza na altura da coxa
Entorse/laceração do ligamento colateral lateral	Tumefação
	Hipersensibilidade à palpação no local da entorse ou laceração
	Marcha alterada
	Dor
	ADM limitada
	Pode desenvolver fraqueza da coxa
Entorse (deficiência) do LCA	Dor leve se houve laceração completa
	Tumefação
	Deformação
	Pode haver uma ADM limitada
	Sensação de instabilidade
	Pode se desenvolver fraqueza da coxa
	Hipermobilidade articular
Entorse do LCP	Geralmente sem “estalo”
	Dor, em especial com a flexão
	Instabilidade
	Pode desenvolver fraqueza da coxa
Tendinite quadricipital ou patelar (joelho do saltador)	Dor às atividades que exigem sustentação do peso corporal, especialmente corridas e saltos
	A extensão resistida produz sintomas
	Palpação dolorosa na inserção tendinosa, acima ou abaixo da patela
	ADMA é dolorosa e ligeiramente reduzida
	Fraqueza
	Tumefação localizada
	Pode desenvolver atrofia e fibrose
	Anormalidades da textura tecidual
Ruptura do tendão quadricipital ou patelar	A laceração do tensãoqudricipital pode mostrar uma solução de continuidade acima da patela nos estágios agudos
	Incapacidade de estender o joelho ativamente

	Dor no local da ruptura
	Tumefação
	Menor capacidade de sustentação do peso corporal
Bursite patelar	Tumefação ou inchaço
	Pode haver um calo
	Hipersensibilidade à palpação
	Pode ser assintomática
	ADM em geral plena, mas existe retração à pressão excessiva em flexão
Bursite anserina (da pata de ganso)	Dor e hipersensibilidade no local das bursas (bolsas)
	Pode ser reproduzida com flexão resistida do quadril, rotação externa ou interna, abdução e com palpação
	Pode haver uma ADMA ligeiramente reduzida
Síndrome da plica	Dor, geralmente ao longo da patela medial
	Tumefação, em especial sobre a superfície súpero-medial da patela
	Pseudobloqueio e estalidos;
Síndrome do atrito do trato iliotibial (TIT)	Dor sobre a parte lateral do joelho, especialmente durante a atividade esportiva
	A dor pode irradiar se para cima ou para baixo
	Retração de coxa lateral/joelho
	Hipersensibilidade à palpação 2 cm acima da articulação
	Pode haver crepitação com a flexão e a extensão do joelho
Lacerações meniscais	Dor na interlinha articular
	Bloqueio
	Saliência
	Impossibilidade de se estender ou flexionar plenamente o joelho
	Tumefação
	Dor à sustentação do peso corporal
	Imageamento positivo para atrofia do quadrípces
	Podem-se ouvir estalidos articulares
Tríade infeliz (O'Donoghue)	Dor
	Impossibilidade de se sustentar o peso corporal
	Tumefação
	ADM reduzida
	Atrofia
	Fraqueza
	Hipersensibilidade à palpação
	Instabilidade
Osteoartrite	Dor, especialmente à sustentação do peso corporal
	Deformidade (em varo, em algo ou em flexão)
	Movimentação limitada
	Pode haver tumefação
	Fraqueza
	Rigidez matinal

	Pode haver instabilidade
	Atrofia do quadríceps
	Crepitação
Osteocondrite dissecante	Dor difusa no joelho
	Ligeiro derrame
	Hipersensibilidade à palpação
	Atrofia no quadríceps
Meniscectomia	Tumefação
	Dor
	ADM limitada
	Fraqueza
	Atrofia
	Dificuldade com a sustentação do peso corporal
Síndrome de dor patelofemoral	Dor à posição sentada prolongada e ao se descer escada
	Tumefação mínima, ou ausente
	Mecanismo de rastreamento alterado
	Crepitação
	Patela com alinhamento inadequado, em geral lateralmente deslocada ou "olhando de soslaio"
	Dor ao redor e abaixo da patela, especialmente à flexão
	Pode haver alguma proeminência
Luxação/subluxação patelar	Se a luxação dor aguda, observa-se dor intensa e impossibilidade de movimentar o joelho
	A luxação é visível
	Podem existir fraturas por avulsão na superfície medial da patela, dado cartilaginoso e lacerações dos tecidos moles mediais
	As subluxações resultam em hipersensibilidade, especialmente na faceta medial e no retináculomedial
	Em geral hipermóvel
Tornozelo	
Patologias	**Sinais e Sintomas**
Entorse/lacerações agudas do tornozelo lateral	Grau I: dor e tumefação da superfície ântero-lateral do maléolo lateral, hipersensibilidade localizada e nenhuma frouxidão
	Grau II: laceração, estalidos na superfície lateral, dor e tumefação, hipersensibilidade, teste da gaveta anterior positivo, e pode acontecer mais ligamentos
	Grau III: grau II mais laceração completa dos ligamentos, tumefação difusa, equimose e aumento da dor com dificuldade na sustentação do peso corporal; pode ser acompanhada de fraturas

Instabilidade crônica do tornozelo lateral	Instabilidade ao se deambular em colinas ou em terreno irregular
	Hipermobilidade
	Dor e tumefação à atividade
	Fraqueza, especialmente dos fibulares
Doença de Sever	Dor no calcanhar durante atividade
	Hipersensibilidade à palpação
	Tendão calcaniano retraída
Entorses sindesmóticas (diástase do tornozelo)	ADM reduzida
	Hipersensibilidade à palpação
	Tumefação local mínima e equimose
	Dificuldade de sustentar o peso corporal
	Palpação dobre a articulação tibiofibular anterior distal é dolorosa
Tendinite do tibial posterior	Hipersensibilidade à palpação na superfície medial do tornozelo
	Dor com pronação passiva e supinaçãoativa
	Dor com a inversão resistida com flexão plantar que pode irradiar-se proximalmente
Tendinite do Aquileu	Dor, especialmente com atividades de sustentação do peso corporal
	Tumefação
	Hipersensibilidade à palpação
	Espessamento do tendão e crepitação
	Dor com DF passiva e FP resistida
Ruptura do tendão-de-Aquiles	“Estalido” doloroso
	Defeito visível (fenda) no tendão de visualizado imediatamente após o início e se a ruptura dor completa
	ADM reduzida
	Incapacidade de permanecer de pé com apoio nos artelhos
	Dificuldade de realizar a flexão plantar
	Tumefação em 1 a 2h
	Equimose
Lacerações do gastrocnêmio	Dor em queimação;
	Hematoma;
	Hipersensibilidade à palpação
	Anormalidade da textura tecidual
	Retração
	Menor capacidade de sustentação do peso corporal
	Os pacientes relatam a sensação de terem recebido um golpe na perna
Síndrome compartimental por esforço crônico	Dor na perna

	Hipersensibilidade
	Pode haver alguma tumefação/ingurgitamento
	A dor é aliviada por repouso
	Se for mais grave, pode causar dormência e formigamento
Síndrome compartimental aguda	Dor progressiva
	Tumefação progressiva que não diminui com repouso
	Plenitude/retração cutânea
	Dormência e formigamento
	Perda da sensibilidade entre o segundo e o terceiro artelhos
	Pulso pedioso diminuído
Fraturas de estresse	Dor
	Hipersensibilidade à palpação com ou sem tumefação
	Dor a atividades de sustentação do peso corporal e a certos movimentos
Fraturas do tornozelo	Dor
	Tumefação
	Elevação da temperatura local
	Deformidade
	Perda da função
	Incapacidade de se sustentar o peso corporal
	Deve ser avaliada a fíbula proximal para se excluir a presença de fratura nesse osso (fratura de Maisonneuve)
Reparo no tendão de Aquiles	Tumefação
	Dor
	Movimentação limitada
	Hipomobilidade
	Fraqueza
	Atrofia
	Propriocepção reduzida
	Função diminuída
	Capacidade de deambulação reduzida
	A DF plena pode ser limitada
Reconstrução dos ligamentos do tornozelo	Tumefação
	Dor
	Movimentação limitada
	Hipomobilidade
	Fraqueza
	Atrofia
	Propriocepção diminuída

	Função reduzida
	Habilidade para deambulação reduzida
Local: Pé	
Patologias	**Sinais e Sintomas**
Fasciite plantar	Dor, especialmente pela manhã, aliviada em 5 a 10 min, mas que piora durante o dia
	Hipersensibilidade à palpação da superfície medial do calcâneo
	Dor com DF e extensão dos artelhos
Síndrome do túnel do Tarso	Sensação de queimação e dor na planta do pé, especialmente à noite e à sustentação do peso corporal
	Dormência/formigamento
	Não costuma haver hipersensibilidade plantar
Neuroma de Morton	Sensação de queimação ou percepção de um choque elétrico
	Dor intermitente, especialmente à sustentação do peso corporal
	Hipersensibilidade à palpação no local do neuroma
	Calo
Artelho da grama artificial	Dor
	Sensibilidade à palpação
	Tumefação na articulação MTF
	Equimoses
	Dificuldade e dor para elevar o pé e realizar a extensão passiva dos artelhos
Pronação anormal	Pode ou não ser dolorosa na superfície plantar
	Podem-se observar outras falhas biomecânicas: antepé em varo, primeiro raio hipermóvel
	Formação de calosidades, especialmente sob a cabeça do segundo metatarso
	Pode resultar em dor nas costa, tendinite do aquileu, esporões, calcâneos, fasciite plantar, canelite, tendinite do tibial posterior e síndrome patelofemoral
Supinação anormal	Pode ou não ser dolorosa na superfície plantar
	Podem-se observar outras falhas biomecânicas: hipomobilidade do primeiro raio em flexão plantar
	Formação de calosidades, especialmente sob a cabeça do quarto e quinto metatarsos

APÊNDICE B – Relação de Sinais e Sintomas e sua representação na RNA.

Sinais e Sintomas	**Representação RNA**
A dor fica localizada ao redor da região inguinal e raramente irradia-se para o joelho	0,067
Rigidez matinal <30 min	0,52
Dor em queimação, em especial à noite e à atividade	0,211
Desvios da marcha	0,148
Retração capsular	0,511
ADM e função limitadas	0,091
Equimose	0,274
Dor ao alongamento passivo e a ADM	0,181
A dor tem início insidioso, localiza-se na parte lateral do quadril, e pode irradiar-se para o dermátomo L5, especialmente ao subirem escadas, à posição de decúbito lateral ou ao se sair de um carro	0,073
A ADM costuma ser completa, porém a adução pode ser limitada	0,046
Dor localizada na crista ilíaca;	0,229
A dor é de inicio gradual ou súbito e acomete a parte anterolateral da coxa	0,061
A dor é em queimação, sendo agravada por adução e flexão forçada do quadril	0,064
Hipoestesia e dormência ocasional	0,349
A ADM em geral é plena, exceto para retração, e pode ser dolorosa no final da amplitude	0,052
Dor local na área da contusão	0,226
A ADM do joelho limitada, especialmente em flexão	0,049
Podem ser palpadas anormalidades na textura tecidual	0,484
Inicio gradual da dor no quadril, na região inguinal, na coxa e no joelho, especialmente com a sustentação do peso corporal	0,373
Abdução, rotação interna e externa reduzidas	0,088
Espasmos musculares	0,277
Atividades funcionais reduzidas	0,115
Dor leve à palpação no quadril, na perna, na coxa e no joelho, nos casos crônicos	0,22
Discrepâncias no comprimento dos MMII (DCP)	0,166
ADM reduzida, especialmente em RI, abdução e flexão	0,103
Marcha de Trendelenburg;	0,388
Abdução reduzida	0,085
Perna mantida em flexão com abdução	0,427
Em geral muito dolorosa após a cirurgia	0,271
Mobilidade e força limitadas	0,397
Dor	0,169
Flexão e extensão do joelho limitadas	0,283

Padrão da marcha alterado	0,409
Pode desenvolver-se fraqueza na altura da coxa	0,439
Hipersensibilidade à palpação no local da entorse ou laceração	0,343
Marcha alterada	0,385
ADM limitada	0,097
Pode desenvolver fraqueza da coxa	0,436
Dor leve se houve laceração completa	0,223
Deformação	0,139
Pode haver uma ADM limitada	0,46
Sensação de instabilidade	0,532
Pode se desenvolver fraqueza da coxa	0,472
Hipermobilidade articular	0,322
Geralmente sem “estalo”	0,304
Dor, em especial com a flexão	0,25
Instabilidade	0,376
Dor às atividades que exigem sustentação do peso corporal, especialmente corridas e saltos	0,187
A extensão resistida produz sintomas	0,076
Palpação dolorosa na inserção tendinosa, acima ou abaixo da patela	0,415
ADMA é dolorosa e ligeiramente reduzida	0,106
Tumefação localizada	0,556
Pode desenvolver atrofia e fibrose	0,433
Anormalidades da textura tecidual	0,109
A laceração do tensãoqudricipital pode mostrar uma solução de continuidade acima da patela nos estágios agudos	0,079
Incapacidade de estender o joelho ativamente	0,364
Dor no local da ruptura	0,241
Menor capacidade de sustentação do peso corporal	0,394
Tumefação ou inchaço	0,565
Pode haver um calo	0,457
Pode ser assintomática	0,475
ADM em geral plena, mas existe retração à pressão excessiva em flexão	0,094
Dor e hipersensibilidade no local das bursas (bolsas)	0,205
Pode ser reproduzida com flexão resistida do quadril, rotação externa ou interna, abdução e com palpação	0,478
Pode haver uma ADMA ligeiramente reduzida	0,463
Dor, geralmente ao longo da patela medial	0,259
Tumefação, em especial sobre a superfície súpero-medial da patela	0,571
Pseudobloqueio e estalidos;	0,502
Dor sobre a parte lateral do joelho, especialmente durante a atividade esportiva	0,247
A dor pode irradiar se para cima ou para baixo	0,07

Retração de coxa lateral/joelho	0,514
Hipersensibilidade à palpação 2 cm acima da articulação	0,331
Pode haver crepitação com a flexão e a extensão do joelho	0,448
Dor na interlinha articular	0,232
Bloqueio	0,124
Saliência	0,523
Impossibilidade de se estender ou flexionar plenamente o joelho	0,358
Dor à sustentação do peso corporal	0,178
Imageamento positivo para atrofia do quadrípces	0,355
Podem-se ouvir estalidos articulares	0,493
Impossibilidade de se sustentar o peso corporal	0,361
Deformidade (em varo, em algo ou em flexão)	0,145
Movimentação limitada	0,4
Pode haver tumefação	0,454
Rigidez matinal	0,517
Pode haver instabilidade	0,451
Crepitação	0,133
Dor difusa no joelho	0,202
Ligeiro derrame	0,382
Atrofia no quadríceps	0,121
Dificuldade com a sustentação do peso corporal	0,154
Dor à posição sentada prolongada e ao se descer escada	0,175
Tumefação mínima, ou ausente	0,559
Mecanismo de rastreamento alterado	0,391
Patela com alinhamento inadequado, em geral lateralmente deslocada ou “olhando de soslaio”	0,418
Dor ao redor e abaixo da patela, especialmente à flexão	0,184
Pode haver alguma proeminência	0,442
Se a luxação dor aguda, observa-se dor intensa e impossibilidade de movimentar o joelho	0,526
A luxação é visível	0,082
Podem existir fraturas por avulsão na superfície medial da patela, dado cartilaginoso e lacerações dos tecidos moles mediais	0,481
As subluxações resultam em hipersensibilidade, especialmente na faceta medial e no retináculomedial	0,112
Em geral hipermóvel	0,268
Grau I: dor e tumefação da superfície ântero-lateral do maléolo lateral, hipersensibilidade localizada e nenhuma frouxidão	0,307
Grau II: laceração, estalidos na superfície lateral, dor e tumefação, hipersensibilidade, teste da gaveta anterior positivo, e pode acontecer mais ligamentos	0,04

Grau III: grau II mais laceração completa dos ligamentos, tumefação difusa, equimose e aumento da dor com dificuldade na sustentação do peso corporal; pode ser acompanhada de fraturas	0,31
Instabilidade ao se deambular em colinas ou em terreno irregular	0,379
Hipermobilidade	0,319
Dor e tumefação à atividade	0,208
Fraqueza, especialmente dos fibulares	0,295
Dor no calcanhar durante atividade	0,238
Tendão calcaniano retraída	0,544
Tumefação local mínima e equimose	0,553
Dificuldade de sustentar o peso corporal	0,16
Palpação dobre a articulação tibiofibular anterior distal é dolorosa	0,412
Hipersensibilidade à palpação na superfície medial do tornozelo	0,34
Dor com pronação passiva e supinaçãoativa	0,199
Dor com a inversão resistida com flexão plantar que pode irradiar-se proximalmente	0,19
Dor, especialmente com atividades de sustentação do peso corporal	0,253
Espessamento do tendão e crepitação	0,28
Dor com DF passiva e FP resistida	0,196
“Estalido” doloroso	0,043
Defeito visível (fenda) no tendão de visualizado imediatamente após o início e se a ruptura dor completa	0,136
ADM reduzida	0,1
Incapacidade de permanecer de pé com apoio nos artelhos	0,367
Dificuldade de realizar a flexão plantar	0,157
Tumefação em 1 a 2h	0,55
Dor em queimação;	0,214
Hematoma;	0,316
Hipersensibilidade à palpação	0,328
Retração	0,508
Os pacientes relatam a sensação de terem recebido um golpe na perna	0,406
Dor na perna	0,235
Hipersensibilidade	0,325
Pode haver alguma tumefação/ingurgitamento	0,445
A dor é aliviada por repouso	0,058
Se for mais grave, pode causar dormência e formigamento	0,529
Dor progressiva	0,244
Tumefação progressiva que não diminui com repouso	0,568
Plenitude/retração cutânea	0,43
Dormência e formigamento	0,262

Perda da sensibilidade entre o segundo e o terceiro artelhos	0,424
Pulso pedioso diminuído	0,505
Hipersensibilidade à palpação com ou sem tumefação	0,334
Dor a atividades de sustentação do peso corporal e a certos movimentos	0,172
Elevação da temperatura local	0,265
Deformidade	0,142
Perda da função	0,421
Incapacidade de se sustentar o peso corporal	0,37
Deve ser avaliada a fíbula proximal para se excluir a presença de fratura nesse osso (fratura de Maisonneuve)	0,151
Hipomobilidade	0,352
Propriocepção reduzida	0,499
Função diminuída	0,298
Capacidade de deambulação reduzida	0,13
A DF plena pode ser limitada	0,055
Tumefação	0,547
Fraqueza	0,292
Atrofia	0,118
Propriocepção diminuída	0,496
Função reduzida	0,301
Habilidade para deambulação reduzida	0,313
Dor, especialmente pela manhã, aliviada em 5 a 10 min, mas que piora durante o dia	0,256
Hipersensibilidade à palpação da superfície medial do calcâneo	0,337
Dor com DF e extensão dos artelhos	0,193
Sensação de queimação e dor na planta do pé, especialmente à noite e à sustentação do peso corporal	0,535
Não costuma haver hipersensibilidade plantar	0,403
Sensação de queimação ou percepção de um choque elétrico	0,538
Dor intermitente, especialmente à sustentação do peso corporal	0,217
Hipersensibilidade à palpação no local do neuroma	0,346
Calo	0,127
Sensibilidade à palpação	0,541
Tumefação na articulação MTF	0,562
Dificuldade e dor para elevar o pé e realizar a extensão passiva dos artelhos	0,163
Pode ou não ser dolorosa na superfície plantar	0,466
Podem-se observar outras falhas biomecânicas: antepé em varo, primeiro raio hipermóvel	0,487
Formação de calosidades, especialmente sob a cabeça do segundo metatarso	0,289

Pode resultar em dor nas costa, tendinite do aquileu, esporões, calcâneos, fasciite plantar, canelite, tendinite do tibial posterior e síndrome patelofemoral	0,469
Podem-se observar outras falhas biomecânicas: hipomobilidade do primeiro raio em flexão plantar	0,49
Formação de calosidades, especialmente sob a cabeça do quarto e quinto metatarsos	0,286

APÊNDICE C – Arquivo com padrões de treinamento.

SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

No. of patterns : 35
No. of input units : 10
No. of output units : 1

Quadril e coxa
Input pattern 1:
0.067 0.52 0.211 0.148 0.511 0.091 0 0 0 0
Output pattern 1:
0.1
Input pattern 2:
0.328 0.547 0.274 0.181 0.109 0 0 0 0 0
Output pattern 2:
0.11
Input pattern 3:
0.073 0.046 0 0 0 0 0 0 0 0
Output pattern 3:
0.12
Input pattern 5:
0.061 0.064 0.349 0.052 0.328 0 0 0 0 0
Output pattern 5:
0.14
Input pattern 7:
0.373 0.328 0.088 0.388 0.277 0.115 0 0 0 0
Output pattern 7:
0.16
Input pattern 8:
0.22 0.166 0.121 0.103 0.388 0 0 0 0 0
Output pattern 8:
0.17
Input pattern 9:
0.085 0.427 0 0 0 0 0 0 0 0
Output pattern 9:
0.18
Joelho
Input pattern 11:
0.169 0.283 0.547 0.343 0.409 0.439 0 0 0 0
Output pattern 11:
0.2
Input pattern 13:
0.223 0.547 0.139 0.46 0.532 0.472 0.322 0 0 0
Output pattern 13:
0.22
Input pattern 14:
0.304 0.25 0.376 0.439 0 0 0 0 0 0
Output pattern 14:
0.23
Input pattern 15:
0.187 0.076 0.415 0.106 0.292 0.556 0.433 0.109 0 0
Output pattern 15:
0.24

Input pattern 16:
0.079 0.364 0.241 0.547 0.394 0 0 0 0 0
Output pattern 16:
0.25
Input pattern 18:
0.205 0.478 0.463 0 0 0 0 0 0 0
Output pattern 18:
0.27
Input pattern 19:
0.259 0.571 0.502 0 0 0 0 0 0 0
Output pattern 19:
0.28
Input pattern 21:
0.232 0.124 0.523 0.358 0.547 0.178 0.355 0.493 0 0
Output pattern 21:
0.3
Input pattern 22:
0.169 0.361 0.547 0.1 0.118 0.292 0.328 0.376 0 0
Output pattern 22:
0.31
Input pattern 23:
0.178 0.145 0.4 0.454 0.292 0.517 0.451 0.121 0.133 0
Output pattern 23:
0.32
Input pattern 25:
0.547 0.169 0.097 0.292 0.118 0.154 0 0 0 0
Output pattern 25:
0.34
Input pattern 26:
0.175 0.559 0.391 0.133 0.418 0.184 0.442 0 0 0
Output pattern 26:
0.35
Input pattern 27:
0.526 0.082 0.481 0.112 0.268 0 0 0 0 0
Output pattern 27:
0.36
Input pattern 28:
0.307 0.04 0.31 0 0 0 0 0 0 0
Output pattern 28:
0.37
Input pattern 29:
0.379 0.319 0.208 0.295 0 0 0 0 0 0
Output pattern 29:
0.38
Input pattern 31:
0.1 0.328 0.553 0.16 0.412 0 0 0 0 0
Output pattern 31:
0.4
Input pattern 32:
0.34 0.199 0.19 0 0 0 0 0 0 0
Output pattern 32:
0.41
Input pattern 33:
0.253 0.547 0.328 0.28 0.196 0 0 0 0 0
Output pattern 33:

0.42
Input pattern 35:
0.214 0.316 0.328 0.109 0.508 0.394 0.406 0 0 0
Output pattern 35:
0.44
Input pattern 36:
0.235 0.325 0.445 0.058 0.529 0 0 0 0 0
Output pattern 36:
0.45
Input pattern 37:
0.244 0.568 0.43 0.262 0.424 0.505 0 0 0 0
Output pattern 37:
0.46
Input pattern 39:
0.169 0.547 0.265 0.142 0.421 0.37 0.151 0 0 0
Output pattern 39:
0.48
Input pattern 40:
0.547 0.169 0.4 0.352 0.292 0.118 0.499 0.298 0.13 0.055
Output pattern 40:
0.49
Input pattern 41:
0.547 0.169 0.4 0.352 0.292 0.118 0.499 0.298 0.313 0
Output pattern 41:
0.5
Pe
Input pattern 42:
0.256 0.337 0.193 0 0 0 0 0 0 0
Output pattern 42:
0.51
Input pattern 44:
0.538 0.217 0.346 0.127 0 0 0 0 0 0
Output pattern 44:
0.53
Input pattern 45:
0.169 0.541 0.562 0.274 0.163 0 0 0 0 0
Output pattern 45:
0.54
Input pattern 47:
0.466 0.49 0.286 0 0 0 0 0 0 0
Output pattern 47:
0.56

APÊNDICE D – Arquivo com padrões de validação.

SNNS pattern definition file V3.2
generated at Mon Apr 25 15:58:23 1994

No. of patterns : 12
No. of input units : 10
No. of output units : 1

Input pattern 4:
0.229 0 0 0 0 0 0 0 0 0
Output pattern 4:
0.13
Input pattern 6:
0.349 0.049 0.484 0 0 0 0 0 0 0
Output pattern 6:
0.15
Input pattern 10:
0.271 0.397 0 0 0 0 0 0 0 0
Output pattern 10:
0.19
Input pattern 12:
0.547 0.343 0.385 0.169 0.097 0.439 0 0 0 0
Output pattern 12:
0.21
Input pattern 17:
0.565 0.457 0.328 0.475 0.094 0 0 0 0 0
Output pattern 17:
0.26
Input pattern 20:
0.247 0.07 0.514 0.331 0.448 0 0 0 0 0
Output pattern 20:
0.29
Input pattern 24:
0.202 0.382 0.328 0.121 0 0 0 0 0 0
Output pattern 24:
0.33
Input pattern 30:
0.238 0.328 0.544 0 0 0 0 0 0 0
Output pattern 30:
0.39
Input pattern 34:
0.043 0.136 0.1 0.367 0.157 0.55 0.274 0 0 0
Output pattern 34:
0.43
Input pattern 38:
0.169 0.334 0.172 0 0 0 0 0 0 0
Output pattern 38:
0.47
Input pattern 43:
0.535 0.262 0.403 0 0 0 0 0 0 0
Output pattern 43:
0.52
Input pattern 46:
0.466 0.487 0.289 0.469 0 0 0 0 0 0

Output pattern 46:
0.55

ANEXOS

ANEXO A – Representação da RNA01 em código C.

```
/*****************************************************
  RNA01.c
  ------------------------------------------------------
  generated at Sat Jun 09 21:29:56 2012
  by snns2c ( Bernward Kett 1995 )
*****************************************************/

#include <math.h>

#define Act_Logistic(sum, bias)  ( (sum+bias<10000.0) ? ( 1.0/(1.0 + exp(-sum-bias) ) ) : 0.0 )
#ifndef NULL
#define NULL (void *)0
#endif

typedef struct UT {
        float act;        /* Activation       */
        float Bias;       /* Bias of the Unit */
        int   NoOfSources; /* Number of predecessor units */
   struct UT   **sources; /* predecessor units */
        float *weights; /* weights from predecessor units */
       } UnitType, *pUnit;

  /* Forward Declaration for all unit types */
  static UnitType Units[13];
  /* Sources definition section */
  static pUnit Sources[] =  {
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units +
10,

Units + 11,

  };

  /* Weigths definition section */
  static float Weights[] =  {
0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,
0.000000,

0.000000,

  };

  /* unit definition section (see also UnitType) */
  static UnitType Units[13] =
  {
    { 0.0, 0.0, 0, NULL , NULL },
    { /* unit 1 (Input) */
      0.0, 0.000000, 0,
       &Sources[0] ,
       &Weights[0] ,
      },
    { /* unit 2 (Input) */
      0.0, 0.000000, 0,
```

```
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 3 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 4 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 5 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 6 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 7 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 8 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 9 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 10 (Input) */
   0.0, 0.000000, 0,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 11 (Hidden) */
   0.0, 0.000000, 10,
    &Sources[0] ,
    &Weights[0] ,
   },
  { /* unit 12 (Output) */
   0.0, 0.000000, 1,
    &Sources[10] ,
    &Weights[10] ,
   }

};
```

```
int RNA01(float *in, float *out, int init)
{
  int member, source;
  float sum;
  enum{OK, Error, Not_Valid};
  pUnit unit;


  /* layer definition section (names & member units) */

   static pUnit Input[10] = {Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7,
Units + 8, Units + 9, Units + 10}; /* members */

  static pUnit Hidden1[1] = {Units + 11}; /* members */

  static pUnit Output1[1] = {Units + 12}; /* members */

  static int Output[1] = {12};

  for(member = 0; member < 10; member++) {
    Input[member]->act = in[member];
  }

  for (member = 0; member < 1; member++) {
    unit = Hidden1[member];
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };

  for (member = 0; member < 1; member++) {
    unit = Output1[member];
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };

  for(member = 0; member < 1; member++) {
    out[member] = Units[Output[member]].act;
  }

  return(OK);
}
```

ANEXO B – Representação da RNA02 em código C.

```
/******************************************************
  RNA02.c
  -------------------------------------------------
  generated at Sat Jun 09 21:30:10 2012
  by snns2c ( Bernward Kett 1995 )
******************************************************/

#include <math.h>

#define Act_Logistic(sum, bias)  ( (sum+bias<10000.0) ? ( 1.0/(1.0 + exp(-sum-bias) ) ) : 0.0 )
#ifndef NULL
#define NULL (void *)0
#endif

typedef struct UT {
        float act;        /* Activation       */
        float Bias;       /* Bias of the Unit */
        int   NoOfSources; /* Number of predecessor units */
   struct UT   **sources; /* predecessor units */
        float *weights; /* weights from predecessor units */
       } UnitType, *pUnit;

  /* Forward Declaration for all unit types */
  static UnitType Units[15];
  /* Sources definition section */
  static pUnit Sources[] =  {
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units +
10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units +
10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units +
10,

Units + 11, Units + 12, Units + 13,

  };

  /* Weigths definition section */
  static float Weights[] =  {
0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,
0.000000,

0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,
0.000000,

0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,
0.000000,

0.000000, 0.000000, 0.000000,
```

```
};

/* unit definition section (see also UnitType) */
static UnitType Units[15] =
{
  { 0.0, 0.0, 0, NULL , NULL },
  { /* unit 1 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 2 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 3 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 4 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 5 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 6 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 7 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 8 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 9 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 10 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
```

```
    },
   { /* unit 11 (Hidden) */
    0.0, 0.000000, 10,
     &Sources[0] ,
     &Weights[0] ,
    },
   { /* unit 12 (Hidden) */
    0.0, 0.000000, 10,
     &Sources[10] ,
     &Weights[10] ,
    },
   { /* unit 13 (Hidden) */
    0.0, 0.000000, 10,
     &Sources[20] ,
     &Weights[20] ,
    },
   { /* unit 14 (Output) */
    0.0, 0.000000, 3,
     &Sources[30] ,
     &Weights[30] ,
    }

  };



int RNA02(float *in, float *out, int init)
{
  int member, source;
  float sum;
  enum{OK, Error, Not_Valid};
  pUnit unit;


  /* layer definition section (names & member units) */

  static pUnit Input[10] = {Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7,
Units + 8, Units + 9, Units + 10}; /* members */

  static pUnit Hidden1[3] = {Units + 11, Units + 12, Units + 13}; /* members */

  static pUnit Output1[1] = {Units + 14}; /* members */

  static int Output[1] = {14};

  for(member = 0; member < 10; member++) {
    Input[member]->act = in[member];
  }

  for (member = 0; member < 3; member++) {
    unit = Hidden1[member];
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
```

```
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };

  for (member = 0; member < 1; member++) {
    unit = Output1[member];
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };

  for(member = 0; member < 1; member++) {
    out[member] = Units[Output[member]].act;
  }

  return(OK);
}
```

ANEXO C – Representação da RNA03 em código C.

```
/******************************************************
  RNA03.c
  -------------------------------------------------------
  generated at Sat Jun 09 21:30:19 2012
  by snns2c ( Bernward Kett 1995 )
******************************************************/

#include <math.h>

#define Act_Logistic(sum, bias)  ( (sum+bias<10000.0) ? ( 1.0/(1.0 + exp(-sum-bias) ) ) : 0.0 )
#ifndef NULL
#define NULL (void *)0
#endif

typedef struct UT {
        float act;        /* Activation       */
        float Bias;       /* Bias of the Unit */
        int   NoOfSources; /* Number of predecessor units */
   struct UT   **sources; /* predecessor units */
        float *weights; /* weights from predecessor units */
       } UnitType, *pUnit;

  /* Forward Declaration for all unit types */
  static UnitType Units[18];
  /* Sources definition section */
  static pUnit Sources[] =  {
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units +
10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units +
10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units + 9, Units +
10,

Units + 11, Units + 12, Units + 13,
Units + 11, Units + 12, Units + 13,
Units + 11, Units + 12, Units + 13,
Units + 14, Units + 15, Units + 16,

  };

  /* Weigths definition section */
  static float Weights[] =  {
0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,
0.000000,

0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,
0.000000,

0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,  0.000000,
0.000000,
```

```
0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000,

};

/* unit definition section (see also UnitType) */
static UnitType Units[18] =
{
  { 0.0, 0.0, 0, NULL , NULL },
  { /* unit 1 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 2 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 3 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 4 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 5 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 6 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 7 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 8 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
  { /* unit 9 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
```

```
    },
   { /* unit 10 (Input) */
    0.0, 0.000000, 0,
     &Sources[0] ,
     &Weights[0] ,
    },
   { /* unit 11 (Hidden) */
    0.0, 0.000000, 10,
     &Sources[0] ,
     &Weights[0] ,
    },
   { /* unit 12 (Hidden) */
    0.0, 0.000000, 10,
     &Sources[10] ,
     &Weights[10] ,
    },
   { /* unit 13 (Hidden) */
    0.0, 0.000000, 10,
     &Sources[20] ,
     &Weights[20] ,
    },
   { /* unit 14 (Hidden) */
    0.0, 0.000000, 3,
     &Sources[30] ,
     &Weights[30] ,
    },
   { /* unit 15 (Hidden) */
    0.0, 0.000000, 3,
     &Sources[33] ,
     &Weights[33] ,
    },
   { /* unit 16 (Hidden) */
    0.0, 0.000000, 3,
     &Sources[36] ,
     &Weights[36] ,
    },
   { /* unit 17 (Output) */
    0.0, 0.000000, 3,
     &Sources[39] ,
     &Weights[39] ,
    }

  };



int RNA03(float *in, float *out, int init)
{
  int member, source;
  float sum;
  enum{OK, Error, Not_Valid};
  pUnit unit;


  /* layer definition section (names & member units) */
```

```
  static pUnit Input[10] = {Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7,
Units + 8, Units + 9, Units + 10}; /* members */

  static pUnit Hidden1[3] = {Units + 11, Units + 12, Units + 13}; /* members */

  static pUnit Hidden2[3] = {Units + 14, Units + 15, Units + 16}; /* members */

  static pUnit Output1[1] = {Units + 17}; /* members */

  static int Output[1] = {17};

  for(member = 0; member < 10; member++) {
   Input[member]->act = in[member];
  }

  for (member = 0; member < 3; member++) {
   unit = Hidden1[member];
   sum = 0.0;
   for (source = 0; source < unit->NoOfSources; source++) {
    sum += unit->sources[source]->act
         * unit->weights[source];
   }
   unit->act = Act_Logistic(sum, unit->Bias);
  };

  for (member = 0; member < 3; member++) {
   unit = Hidden2[member];
   sum = 0.0;
   for (source = 0; source < unit->NoOfSources; source++) {
    sum += unit->sources[source]->act
         * unit->weights[source];
   }
   unit->act = Act_Logistic(sum, unit->Bias);
  };

  for (member = 0; member < 1; member++) {
   unit = Output1[member];
   sum = 0.0;
   for (source = 0; source < unit->NoOfSources; source++) {
    sum += unit->sources[source]->act
         * unit->weights[source];
   }
   unit->act = Act_Logistic(sum, unit->Bias);
  };

  for(member = 0; member < 1; member++) {
   out[member] = Units[Output[member]].act;
  }

  return(OK);
}
```

ANEXO D – Representação da RNA04 em código C.

```
/*********************************************************
  RNA04.c
  --------------------------------------------------------
  generated at Sat Jun 09 21:30:31 2012
  by snns2c ( Bernward Kett 1995 )
*********************************************************/

#include <math.h>

#define Act_Logistic(sum, bias)  ( (sum+bias<10000.0) ? ( 1.0/(1.0 + exp(-sum-bias) ) ) : 0.0
)
#ifndef NULL
#define NULL (void *)0
#endif

typedef struct UT {
        float act;        /* Activation       */
        float Bias;       /* Bias of the Unit */
        int   NoOfSources; /* Number of predecessor units */
  struct UT   **sources; /* predecessor units */
        float *weights; /* weights from predecessor units */
      } UnitType, *pUnit;

  /* Forward Declaration for all unit types */
  static UnitType Units[24];
  /* Sources definition section */
  static pUnit Sources[] =  {
Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units +
9, Units + 10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units +
9, Units + 10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units +
9, Units + 10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units +
9, Units + 10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units +
9, Units + 10,

Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6, Units + 7, Units + 8, Units +
9, Units + 10,

Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
```

```
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units + 16,
Units + 17, Units + 18, Units + 19, Units + 20, Units + 21, Units + 22,

  };

  /* Weigths definition section */
  static float Weights[] =  {
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000,

0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000,

0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000,

0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000,

0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000,

0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000,

0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000,

  };

  /* unit definition section (see also UnitType) */
  static UnitType Units[24] =
  {
    { 0.0, 0.0, 0, NULL , NULL },
    { /* unit 1 (Input) */
      0.0, 0.000000, 0,
       &Sources[0] ,
       &Weights[0] ,
      },
    { /* unit 2 (Input) */
      0.0, 0.000000, 0,
       &Sources[0] ,
       &Weights[0] ,
```

```
  },
{ /* unit 3 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 4 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 5 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 6 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 7 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 8 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 9 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 10 (Input) */
  0.0, 0.000000, 0,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 11 (Hidden) */
  0.0, 0.000000, 10,
   &Sources[0] ,
   &Weights[0] ,
  },
{ /* unit 12 (Hidden) */
  0.0, 0.000000, 10,
   &Sources[10] ,
   &Weights[10] ,
```

```
 },
{ /* unit 13 (Hidden) */
 0.0, 0.000000, 10,
  &Sources[20] ,
  &Weights[20] ,
 },
{ /* unit 14 (Hidden) */
 0.0, 0.000000, 10,
  &Sources[30] ,
  &Weights[30] ,
 },
{ /* unit 15 (Hidden) */
 0.0, 0.000000, 10,
  &Sources[40] ,
  &Weights[40] ,
 },
{ /* unit 16 (Hidden) */
 0.0, 0.000000, 10,
  &Sources[50] ,
  &Weights[50] ,
 },
{ /* unit 17 (Hidden) */
 0.0, 0.000000, 6,
  &Sources[60] ,
  &Weights[60] ,
 },
{ /* unit 18 (Hidden) */
 0.0, 0.000000, 6,
  &Sources[66] ,
  &Weights[66] ,
 },
{ /* unit 19 (Hidden) */
 0.0, 0.000000, 6,
  &Sources[72] ,
  &Weights[72] ,
 },
{ /* unit 20 (Hidden) */
 0.0, 0.000000, 6,
  &Sources[78] ,
  &Weights[78] ,
 },
{ /* unit 21 (Hidden) */
 0.0, 0.000000, 6,
  &Sources[84] ,
  &Weights[84] ,
 },
{ /* unit 22 (Hidden) */
 0.0, 0.000000, 6,
  &Sources[90] ,
  &Weights[90] ,
```

```
    },
   { /* unit 23 (Output) */
    0.0, 0.000000, 6,
     &Sources[96] ,
     &Weights[96] ,
    }

 };



int RNA04(float *in, float *out, int init)
{
 int member, source;
 float sum;
 enum{OK, Error, Not_Valid};
 pUnit unit;


 /* layer definition section (names & member units) */

  static pUnit Input[10] = {Units + 1, Units + 2, Units + 3, Units + 4, Units + 5, Units + 6,
Units + 7, Units + 8, Units + 9, Units + 10}; /* members */

 static pUnit Hidden1[6] = {Units + 11, Units + 12, Units + 13, Units + 14, Units + 15, Units
+ 16}; /* members */

 static pUnit Hidden2[6] = {Units + 17, Units + 18, Units + 19, Units + 20, Units + 21, Units
+ 22}; /* members */

 static pUnit Output1[1] = {Units + 23}; /* members */

 static int Output[1] = {23};

 for(member = 0; member < 10; member++) {
  Input[member]->act = in[member];
 }

 for (member = 0; member < 6; member++) {
  unit = Hidden1[member];
  sum = 0.0;
  for (source = 0; source < unit->NoOfSources; source++) {
   sum += unit->sources[source]->act
        * unit->weights[source];
  }
  unit->act = Act_Logistic(sum, unit->Bias);
 };

 for (member = 0; member < 6; member++) {
  unit = Hidden2[member];
```

```
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };

  for (member = 0; member < 1; member++) {
    unit = Output1[member];
    sum = 0.0;
    for (source = 0; source < unit->NoOfSources; source++) {
      sum += unit->sources[source]->act
            * unit->weights[source];
    }
    unit->act = Act_Logistic(sum, unit->Bias);
  };

  for(member = 0; member < 1; member++) {
    out[member] = Units[Output[member]].act;
  }

  return(OK);
}
```