Instituto Superior de Tecnologia em Ciência da Computação de Petrópolis - ISTCC-P

FAETEC

Daniel Tré da Conceição

Redes Neurais Artificiais Aplicadas ao Jogo da Velha 3D em Pinos

Petrópolis - RJ

2009

Daniel Tré da Conceição

Redes Neurais Artificiais Aplicadas ao Jogo da Velha 3D em Pinos

Trabalho de conclusão apresentado ao Curso de graduação Tecnólogo da Informação e da Comunicação, Instituto Superior de Tecnologia em Ciência da Computação de Petrópolis - ISTCC-P, como requisito parcial para obtenção do grau de Tecnólogo da Informação e da Comunicação.

Orientador: Profº . Fábio Borges de Oliveira

Petrópolis - RJ

2009

Daniel Tré da Conceição

Redes Neurais Artificiais Aplicadas ao Jogo da Velha 3D em Pinos

Trabalho de conclusão de curso de graduação apresentado ao Instituto Superior de Tecnologia em Ciências da Computação de Petrópolis como parte dos requisitos para obtenção do título de "Tecnólogo da Informação e da Comunicação".

Fábio Borges de Oliveira, M. Sci.
Orientador IST/LNCC

Saul de Castro Leite, D. Sci.
LNCC

Daniele Quintella Mendes Madureira, D. Sci.
LNCC

Ana Cristina Gomes Silveira, B. Sci.
LNCC

Petrópolis, RJ
2009

RESUMO

O interesse pelo ramo de redes neurais artificiais deve-se ao potencial que elas podem alcançar quando bem elaboradas, bem treinadas e bem aplicadas. Os objetivos deste trabalho são entender como uma aplicação com redes neurais funciona e aplicar a rede neural especificamente em um Jogo da Velha 3D em Pinos. Este trabalho está dividido em duas partes: Na primeira são descritas as redes neurais artificiais para que se possa descrever teoricamente como criar, treinar e utilizar uma rede neural artificial. Na segunda parte descreve-se, cria-se e treina-se uma rede neural artificial para jogar o Jogo da Velha 3D em Pinos. É aplicada na tomada de decisões uma rede FeedForward e para treinamento utiliza-se a aprendizagem por correção de erro. Para agilizar o treinamento são criadas quatro redes para que possam treinar jogando uma contra as outras e um juiz para validar e classificar as jogadas. O juiz faz o papel do professor para as redes, informando qual a melhor jogada em cada situação.

Palavras-chave:
Redes Neurais Artificiais, Jogo da Velha 3D em Pinos

ABSTRACT

Interest in the field of artificial neural networks is attributed to the potential they can have when well developed, well trained and well implemented. The objectives of this study are to understand how an application works with neural networks and to apply the artificial neural network specifically in a Tic Tac Toe 3D Pin. This study is divided into two parts: The first part describes the neural networks to allow for a better understanding about to create, train and use an artificial neural network. The second part describes, creates and trains artificial neural network to play Tic Tac Toe 3D Pin game, for this is applied in the decision-making a feedforward neural network and for the training is used the error correction learning. To speed up the training, we created four artificial neural networks so they can train together and they play one against the other and a judge to validate, classify and teach the moves.

Keywords:
Artificial Neural Networks, Tic Tac Toe 3D pin

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

INTRODUÇÃO

A motivação para o desenvolvimento de pesquisas na área de redes neurais artificiais deve-se ao fato de que o cérebro humano processa as informações de forma completamente diferente dos computadores digitais convencionais. Para o desenvolvimento de redes neurais artificiais buscou-se o conhecimento nas áreas de ciência cognitiva, cibernética, psicologia, neurobiologia, matemática e física.

Os trabalhos e estudos com redes neurais artificiais são inspirados na estrutura neural biológica, atualmente não há redes neurais artificiais que consigam executar os processamentos do cérebro como um todo, pois a maioria delas são específicas, isto quer dizer que seu trabalho é limitado a uma determinada tarefa, já o cérebro pode executar várias tarefas ao mesmo tempo. Atualmente não se tem subsídios para imitar o funcionamento do cérebro, pois ele é um emaranhando de neurônios ligados uns aos outros, e o aumento e diminuição do número de neurônios e ligações sinápticas são dinâmicos e em tempo real. Os cientistas já o mapearam e sabem quais áreas do cérebro são responsáveis pela maioria das nossas funções (motora, visual, auditivo, etc), isso nos proporciona a raciocinar sobre as redes neurais artificias atuais que apesar de específicas, se assemelham a essas áreas do cérebro. Pode-se concluir que o cérebro é formado por diversas redes específicas interconectadas, e em seu todo temos uma rede que forma um funcionamento complexo a partir de simples neurônios conectados. O neurônio é a menor unidade de processamento de informação de uma rede neural (HAYKIN, 2001). Ele é de cinco a seis ordens de grandeza mais lento que as portas lógicas em silício utilizadas nos processadores de computadores: enquanto os eventos em um circuito de silício acontecem na ordem de nanossegundos ($10^{-9}s$), os eventos neurais acontecem na ordem de milissegundos ($10^{-3}s$). Porém o cérebro é uma estrutura com aproximadamente 10 bilhões de neurônios maciçamente ligados uns aos outros, cerca de 60 trilhões de conexões no total (SHEPHERD; KOCH, 1990). Desta forma ele compensa essa lentidão de seus neurônios, com um forte processamento paralelamente distribuído e auto-organização de suas ligações sinápticas, potencializando seu processamento como um todo. Outra questão é a eficiência energética do cérebro, que é de aproximadamente 10 ordens de grandeza mais econômico energeticamente do que a de um processador, cada operação do cérebro consome aproximadamente $10^{-16}J/s$ (Joules por segundo), enquanto cada operação de um processador consome aproximadamente $10^{-6}J/s$

(Joules por segundo) (FAGGIN; MEAD, 1990). Veja a Tabela 1 comparando estas características entre o cérebro e o computador.

Parâmetro	Cérebro	Computador
Material	Orgânico	Metal e plástico
Velocidade	Milisegundos	Nanosegundos
Processamento	Paralelo	Seqüencial
Armazenamento	Adaptativo	Estático
Controle de Processos	Distribuído	Centralizado
Elementos processados	10^{11} a 10^{14}	10^{5} a 10^{6}
Ligações entre elementos	10.000	< 10
Consumo energético	$10^{-16}J/s$	$10^{-6}J/s$

Tabela 1: Computador x Cérebro

O cérebro humano tem bilhões de neurônios, trilhões de conexões e uma estrutura que aprende a se organizar com a experiencia, mas onde estão armazenadas as informações das experiências adquiridas? Espalhadas por toda ela. Cada conexão entre um neurônio possui um nível de excitação ou inibição. Quanto maior a excitação, mais importância tem essa conexão para o neurônio que recebe o estímulo por essa conexão. Essa excitação sozinha não tem valor algum, mas é utilizada ao longo do percurso de um fluxo de informação. Um fluxo inicia quando a rede é estimulada e termina quando ela responde ao estímulo (HAYKIN, 2001). Grosseiramente falando, o neurônio une cada estímulo de excitação ou inibição feito a ele, ao término da união, o neurônio ou se excita ou se inibe, disparando ou não estímulos para os próximos neurônios. Em neurônios artificiais essa excitação ou inibição se dá através de uma função de ativação: ela recebe a união dos estímulos multiplicados pelos pesos das conexões e o resultado da função de ativação faz com que o neurônio seja ativado ou não naquele instante. Assim entendemos como a rede armazena as informações das experiências adquiridas, agora precisamos saber como ela adquire essas informações das experiências, ou melhor dizendo, como uma rede neural aprende. Veja a Tabela 2 comparando um computador a um neurocomputador (TAFNER; XEREZ, 1995).

Foi explicado anteriormente que os pesos das conexões são os graus de afinidade entre um neurônio e outro, então se uma rede responde de forma errada a um estímulo os pesos das conexões são ajustados para que ela tente responder de forma correta da próxima vez. Quando uma rede responde de forma correta a um estímulo, as conexões que ativaram os neurônios são fortalecidas (aumento do peso, ou grau de afinidade) e as que não ativaram são enfraquecidas (diminuição do peso). Da mesma forma que quando a rede responde de forma errada a um estímulo, as conexões que ativaram os neurônios são enfraquecidas (diminuição do peso) e

Computador	Neurocomputador
Executa programas	Aprende
Executa operações lógicas	Executa operações não lógicas, transformações, comparações
Depende do modelo ou do programador	Descobre as relações ou regras dos dados e exemplos
Testa uma hipótese por vez	Testa todas as possibilidades em paralelo

Tabela 2: Computador x Neurocomputador

as que não ativaram são fortalecidas (aumento do peso). Em redes neurais artificiais existem algoritmos desenvolvidos que efetuam as correções dos pesos, que simulam essa situação do peso que acontece no cérebro.

As redes aprendem três metodologias diferentes, com um "professor", com um "crítico"ou por si só. A primeira é a mais simples de entender, a rede adquire os conhecimentos do professor. Na segunda, a rede aprende sozinha, mas com um direcionamento feito pelo crítico que observa a seqüência de estímulos e respostas anteriores e as qualifica, assim, o crítico consegue conduzir a rede ao aprendizado. A terceira forma faz com que a rede aprenda classificar e agrupar padrões por observação aos estímulos.

As rede neurais artificiais podem ser implementadas através de software, hardware e meios ópticos. Em software são as mais comuns pois são mais baratas e rápidas de serem construídas, contudo são mais lentas em sua velocidade de processamento pela necessidade de emulação. Em hardware o desempenho é alto tanto para solucionar problemas de grande volume de processamento quanto para aplicabilidade em softwares de alta criticidade que não podem parar de funcionar, como os de funcionamento em tempo real. Isso se faz potencialmente rápido na computação em larga escala, tornando-a aplicável à tecnologia VLSI (Very-large-scale-integration) que fornece um meio de capturar comportamentos realmente complexos de uma forma altamente hierárquica (MEAD, 1996a). Contudo quando um neurônio tem milhares de conexões torna-se quase impossível implementá-las em hardware por falta de espaço físico. As implementações em meio ótico solucionam os problemas que ocorrem em hardware, pois as conexões são armazenadas em cristais por um sistema holográfico, ou seja, os pesos são armazenados em um padrão de cargas que é formado em um cristal fotorefratário ou em uma superfície fotorefratora quando são expostos à luz (LOESCH, 1996).

A redes neurais artificiais são utilizadas em muitas áreas (LOESCH, 1996):

- Biologia - Compreensão da estrutura do cérebro para explicar funções e comportamentos, memória e revocação, estudo de desarranjos tais como epilepsia, comportamentos como

abuso de drogas por exemplo;

- Engenharia - Construção de melhores sistemas, controle motor na robótica, estabilidade postural em estruturas flexíveis;
- Ciências da computação - Compreensão do processamento do cérebro para construção de computadores e controladores mais rápidos, eficientes e tolerantes a falhas, na resolução de problemas complexos como a recuperação de informação a partir de vestígios parcialmente corretos e na identificação de rostos com variações como luz e expressão facial.

As vantagens da aplicação de redes neurais artificias são imensas pois elas são capazes de efetuar tarefas que um programa convencional não é capaz; podem continuar a responder de forma correta mesmo que um neurônio artificial falhe; aprendem a fazer as tarefas não precisando ser reprogramada; podem ser implementada em qualquer aplicação e sem conhecimento total do problema. E as desvantagens são que ela precisa ser treinada e as vezes é difícil treiná-la; a arquitetura de uma rede neural é diferente da arquitetura de microprocessadores, por isso, é preciso emulá-la, e isto, faz com que haja um elevado tempo de processamento para as grandes redes neurais, além de não ser claro o resultado apresentado fornecido pela rede.

O objetivo teste trabalho é apresentar uma rede neural artificial jogando o Jogo da Velha 3D em Pinos, e para o entendimento é explicado teoricamente, mas sem objetivos profundos matemáticos, o que são as redes neurais artificiais e como são construídas e treinadas.

Este trabalho se configura numa pesquisa teórico-prática, por se partir da teoria das redes neurais, organizando uma aplicação prática no Jogo da Velha 3D em Pinos, visando testar a aprendizagem das redes neurais artificiais e sua habilidade.

Na estruturação da aplicação do Jogo da Velha 3D em Pinos foi usado o programa MatLab por permitir o uso de uma série de biliotecas que têm diversas aplicações em redes neurais que favoreceram o fator tempo na construção do jogo.

1 HISTÓRICO

Em 1943, Warren MuCulloch e Walter Pitts publicaram artigos (MCCULLOCH; PITTS, 1943) que sugeriam a construção de uma máquina baseada no cérebro humano. Em 1949 Donald Hebb escreveu o livro "The Organization of Behavior" (HEBB, 1949a) que abordava idéias antigas e novas relativas ao comportamento psicológico clássico que é o estudo de uma resposta a um estímulo específico e considerava essa uma propriedade de neurônios individuais. O Snark surgia em 1951 como o primeiro neuro computador, construído por Mavin Minsky enquanto trabalhava em Princeton era capaz de processar informações de pouca relevância, serviu como inspiração para as idéias de estruturas que o sucederam.

Em 1956 os pesquisadores da inteligência artificial se depararam com uma bifurcação em seus estudos: simular o comportamento inteligente humano ignorando seus reais mecanismos, ou simular a estrutura do cérebro e seus mecanismos para chegar à inteligência. Essa divisão gerou dois paradigmas e foi chamada no "Darthmouth College" respectivamente de Inteligência Artificial Simbólica e Inteligência Artificial Conexionista.

O Mark I Perceptron surgiu em 1958 como o primeiro neuro computador a obter sucesso, pois ele podia resolver problemas de classificação linearmente separáveis dado um conjunto de elementos. Ele foi criado na Universidade de Cornell por Frank Rosenblatt. Suas técnicas e idéias modernas fizeram com que muitos o vissem como o fundador da neuro computação na forma como a temos hoje (ROSENBLATT, 1958).

Em 1960 Bernard Widrow com a ajuda de alguns estudantes, desenvolveram um novo tipo de elemento de processamento de redes neurais denominado Adaline. Essas redes tem como principais aplicações a filtragem de sinal adaptativo e a equalização adaptativa. São rápidas e fáceis de implementar tanto em circuitos analógicos como em VLSI, contudo, como sua saída é linear somente é possível classificar espaços linearmente separáveis (LOESCH, 1996).

A Adaptive Resonance Theory conhecida como ART utiliza uma grande quantidade de neurônios onde todas possíveis conexões são feitas entre todos os nós, essa grande quantidade conduz alguns neurônios a ficarem ativos e os outros a ficarem inativos. Ela foi publicada em

1983 por G. Carpenter e S. Grossberg (CARPENTER; GROSSBERG, 1983), sua aplicação básica é reconhecimento de padrões sendo capaz de aprender novos padrões sem esquecer os padrões já aprendidos.

A rede do tipo Backpropagation Perceptron tem como característica principal a correção dos pesos sinápticos com base na entrada, a resposta da rede e a resposta desejada. Teve suas publicações entre 1974 e 1986 com P.J. Werbos (WERBOS, 1974) e D. Rumelhart (RUMELHART; HINTON; WILLIAMS, 1986), utilizada principalmente para reconhecimento de padrões, filtragem de sinal, remoção de ruído, segmentação de sinal/imagem, classificação, mapeamento, controle robótico adaptativo, compressão de dados. Seu treinamento é um ponto negativo pois necessita de muito tempo.

Nas redes do tipo Recurrent há uma matriz de pesos sinápticos para cada camada de todos os outros neurônios na rede, não somente os da camada anterior, isso faz com que a complexidade da rede seja bastante alta. Publicada em 1987 por Pineda (PINEDA, 1987) tem como aplicações principais controle robótico, reconhecimento de fala e previsão do elemento sequencial. Esta rede é complexa, e sendo assim, pode ser difícil treiná-la e otimizá-la

Publicada em 1987 por D.W. Tank e J.J. Hopfield (TANK; HOPFIELD, 1987), as redes Time-Delay são utilizadas para reconhecimento de fala e tem o desempenho equivalente aos melhores métodos convencionais para o mesmo fim, sua desvantagem são: janela fixada para a atividade temporal representada, resposta desastrada para diferenças em escala na entrada.

Y.H Pao desenvolveu e publicou uma rede em 1989 (PAO, 1989) chamada de Rede de Ligações Funcionais, ela foi desenvolvida para classificação e mapeamento e com somente duas camadas, uma de entrada e uma de saída ela executa este trabalho sem a necessidade de longo tempo de treino, contudo são difíceis de implementar pela não clareza na forma de identificar as funções adotadas para "Ligações Funcionais".

Em 1974 P.J. Werbos publicou (WERBOS, 1974) a rede Backpropagation de função de utilidade no tempo, utilizada na maximização de utilidade no tempo e neurocontrole na robótica. Ela tem uma abordagem neural mais compreensiva para modelos do controle ou previsão. Suas desvantagens são: pode ser usada somente após a identificação do modelo diferenciável, adaptação off-line se o modelo é dinâmico, e é assumida a exatidão do modelo.

Em 1987 B. Kosko publicou (KOSKO, 1987) a rede BAM (Memória Associativa Bidirecional), utilizada em aplicações de memória endereçada por conteúdo (heteroassociativas). Redes deste tipo são simples, têm regras de aprendizado e sua arquitetura e dinâmica são claras, contudo é pobre sua capacidade de armazenamento e precisão de recuperação.

Na rede do tipo Boltzmann Machine ou Cauchy Machine, os pesos das sinapses de um neurônio são atualizados considerando somente os neurônios ao redor, se assemelhando assim com as redes biológicas. Com publicações em 1984 (HINTON; SEJNOWSKI; ACKLEY, 1984) e 1986 (HINTON; SEJNOWSKI, 1986) (SZU, 1986) e desenvolvida por G. Hinton, T. Sejnowski, D. Ackley, H. Szu, ela é utilizada para reconhecimento de padrões (imagens, sonar, radar), ela é capaz de formar representação ótima das características dos padrões e segue a superfície de energia para obter otimização no ponto mínimo. A máquina de Boltzmann possui tempo de aprendizado mais longo que a máquina de Cauchy.

Publicada em 1977 por J. Anderson (ANDERSON et al., 1977), as redes Brain-State-in-a-Box (BSB), são utilizadas para revocação auto-associativa, ela possivelmente tem melhor desempenho que a rede Hopfield, esta incerteza vem da incompleta exploração em termos de desempenho e aplicabilidade.

Em 1982 é publicada por J. Hopfield (HOPFIELD, 1982) a rede que leva seu próprio sobrenome. A rede consiste de um conjunto de neurônios e um correspondente conjunto de unidades de atraso, formando um sistema de realimentação múltiplo. Ela é utilizada para evocação auto-associativa e otimização. Suas principais vantagens são a conceituação simples, a estabilidade dinâmica e a facilidade de implementação em circuitos VLSI; e as principais desvantagens são a incapacidade de aprender novos estados (pesos fixados no caso da Hopfield discreta), armazenamento de memória pobre e, a estabilidade da rede em estados espúrios.

Com publicações entre 1975 e 1982 K. Fukushima (FUKUSHIMA, 1975) (FUKUSHIMA, 1980) (FUKUSHIMA; MIYAKE, 1982) (FUKUSHIMA; MIYAKE; TAKAYUKE, 1983) (FUKUSHIMA, 1988) desenvolveu uma rede capaz de fazer o reconhecimento de caracteres manuscritos e outras figuras independente da escala, mas ela requer um numero excessivo de neurônios, sua estrutura é complexa e a medida de escala para palavras é um problema que deve ser resolvido.

T. Kohonen publicou em 1981 (KOHONEN, 1981) a rede do tipo Mapas de Preservação da Topologia de Auto-Organização, é utilizada em mapeamentos complexos (envolvendo relações de vizinhança), compreensão de dados e otimização. As desvantagens estão nas características não resolvidas na seleção do número de vetores usados e tempo de treinamento apropriado, além de seu treinamento requerer muito tempo.

Neste trabalho foi utilizada a rede Backpropagation Perceptron no desenvolvimento da aplicação do Jogo da Velha 3D, ela foi escolhida por ser simples de ser implementada, e pelo fato de sua aprendizagem ser feita comparando a resposta da rede com a resposta desejada, além de ser, sem dúvida, o método mais bem difundido e um dos mais bem sucedidos de treinamento de uma rede neural.

2 CONCEITUALIZAÇÃO DE REDE NEURAL ARTIFICIAL

Para alcançar os objetivos deste trabalho há a necessidade de se criar um conceito sobre rede neural artificial. A criação desse conceito se dá através do conhecimento de suas capacidades e propriedades, de sua organização estrutural, de seu aprendizado e de seu processamento.

Segundo (HAYKIN, 2001) uma Rede Neural Artificial é um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples (neurônios) conectadas (sinapses) umas às outras, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos:

- O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem (HAYKIN, 2001).
- Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido (HAYKIN, 2001).

Um neurônio é uma unidade de processamento de informação que é fundamental para a operação de uma rede neural (HAYKIN, 2001).

Uma sinapse é um elo de conexão entre um neurônio e outro. Cada entrada possui uma força ou peso que representa o quanto a entrada é importante para o neurônio (HAYKIN, 2001).

2.1 CAPACIDADES E PROPRIEDADES

Conforme (HAYKIN, 2001) as Redes Neurais Artificiais dispõem de certas capacidades e certas propriedades que as tornam úteis no processamento de dados. As capacidades permitem a rede aprender a responder de forma aceitável e, as propriedades permitem a ela armazenar o conhecimento aprendido, ser modular, ser aplicável a VLSI e ter fundamentação na neurobiologia.

2.1.1 NÃO-LINEARIDADE

É a capacidade das redes neurais artificiais de estabelecer relações não lineares entre os dados de entrada, por exemplo, as vozes das pessoas: Diversas pessoas podem dizer a mesma palavra, contudo umas falarão mais rapidamente, outras mais gravemente, outras gaguejarão, mas mesmo assim a palavra será entendida.

2.1.2 MAPEAMENTO DE ENTRADA-SAÍDA

É a capacidade da rede neural artificial gerar uma saída para cada entrada, uma rede destreinada pode ser moldada às necessidades a partir de treinamento. Uma rede neural artificial aprende o que ensinamos. Seu conhecimento está contido nos seus neurônios e em suas conexões. Cada conexão carrega um peso, e a alteração correta desses pesos faz com que a rede aprenda. Então, para darmos conhecimento a uma rede neural, devemos alterar os pesos das ligações, e fazemos isso treinando-a. Seu treino se dá comumente utilizando técnicas de feedbackward (utiliza-se a resposta da rede como insumo para a correção de seus pesos) com operações da física, da engenharia e da estatística. Isso quer dizer que, dado um conjunto de entradas aplicadas à rede, teremos um conjunto de saídas da rede e um conjunto de saídas desejadas. As operações servem para achar uma aproximação entre as saídas da rede e as saídas desejadas, e, do resultado das operações, obtém-se um valor que será utilizado para alterar os pesos das ligações.

2.1.3 ADAPTABILIDADE

É a capacidade que a rede neural artificial madura tem para poder ser retreinada ou até mesmo se auto-ajustar ao ambiente. Existem dois modos de adaptação para uma rede neural. O primeiro se baseia em um novo treinamento ajustando assim os pesos e alcançando um novo objetivo. O segundo se baseia em a rede alterar seus próprios pesos fazendo assim com que haja adaptabilidade em tempo real. A vantagem de se ter uma rede de adaptabilidade em tempo real é que ela supostamente não será pega de surpresa ao operar em um ambiente não-estacionário. Se a rede conseguir se manter estável em um ambiente não-estacionário ela será dita robusta. A desvantagem acontece quando a rede opera pouco tempo em diversos ambientes diferentes levando-a a responder a perturbações espúrias e tornando-a instável. São buscadas duas propriedades para a adaptabilidade:

- Estabilidade - significa que um padrão de entrada sempre deve ser classificado na mesma

classe ao longo do processo de treinamento;

- Plasticidade - significa que uma rede deve ser capaz de incorporar novo conhecimento sem afetar negativamente o que já foi aprendido.

O Dilema da Estabilidade-Plasticidade consiste no problema que existe para que um sistema de aprendizado mantenha a estabilidade e a plasticidade simultaneamente (GROSSBERG, 1976).

2.1.4 RESPOSTA A EVIDÊNCIAS

É a capacidade da rede neural artificial de responder por aproximação, e trata da crença ou confiança que a rede informa sobre as decisões tomadas. Essa crença faz com que a rede consiga rejeitar padrões ambíguos ao fazer a classificação de padrões. Essa ação também faz com que haja uma melhora no desempenho de classificação.

2.1.5 INFORMAÇÃO CONTEXTUAL

É a propriedade da rede neural artificial manter o conhecimento adquirido distribuído entre seus elementos internos. A quantidade de neurônios, sua estrutura organizacional, os pesos das conexões formam esse conhecimento adquirido. Quando há uma correção dos pesos para que a rede melhore seu aprendizado, cada conexão entre os neurônios da rede é potencialmente afetada pela atividade de todas as outras conexões.

2.1.6 TOLERÂNCIA A FALHAS

É a propriedade da rede neural artificial se manter uma rede funcionando mesmo que um de seus neurônios esteja inativo. Uma rede neural artificial implementada em hardware com milhões de neurônios pode sofrer danos em alguns neurônios sem que a rede seja prejudicada. Isso graças à natureza paralelamente distribuída da informação armazenada. Uma forma de garantir que uma rede neural seja tolerante a falhas é utilizar algoritmos de treinamento que façam uma boa distribuição das informações armazenadas.

2.1.7 IMPLEMENTAÇÃO EM VLSI

É a propriedade da rede neural artificial que a faz ser extremamente adequada para implementações em chips e placas. As redes neurais tem sua natureza maciçamente paralela assim como os chips e placas existentes.

2.1.8 UNIFORMIDADE DE ANÁLISE E PROJETO

É a propriedade da rede neural artificial que permite construir redes modulares através de uma integração homogênea de módulos, isto quer dizer que podemos criar redes específicas e conectá-las criando assim uma rede complexa. Todos os projeto de redes neurais têm a mesma notação em todos os domínios envolvendo sua aplicação de redes. Essa notação é a mesma pois os neurônios são elementos simples e sempre estão presentes em redes neurais. As teorias e os algoritmos de aprendizagem são compartilhadas em aplicações de diferentes redes neurais.

2.1.9 ANALOGIA NEUROBIOLÓGICA

É a propriedade da rede neural artificial em buscar nas redes neurais biológicas suas características, seus comportamentos e sua lógica de processamento e armazenamento de informações.

Um ponto agravante para a utilização de redes neurais é o não conhecimento prévio que ela tem sobre o assunto que se deseja que ela aprenda, com exceção às redes de Hopfield para problemas de otimização, é muito difícil introduzir conhecimento prévio em uma rede, e isso se deve à forma de representação da informação que é maciçamente paralelamente distribuída. Essa distribuição da informação faz com que seja altamente difícil analisar o seu conhecimento e obter cadeias de inferência lógica ou percorrer o procedimento feito pela rede para chegar a uma solução, assim como é possível fazer com os componentes de explicação (passo a passo das operações efetudas para a tomada de uma decisão que normalmente segue a lógica de quem fez a programação do componente) de sistemas especialistas.

2.2 ESTRUTURA

A estrutura define a forma como cada neurônio processa a informação, como ele se conecta a outros neurônios e como uma rede se comunica com outra.

2.2.1 MICRO-ESTRUTURA

A Micro-Estrutura define quais serão as características de cada neurônio de uma rede neural artificial. O neurônio é a menor unidade de processamento de informação e é fundamental para a operação de uma rede neural. Um neurônio artificial é descrito conforme a Figura 2.1.

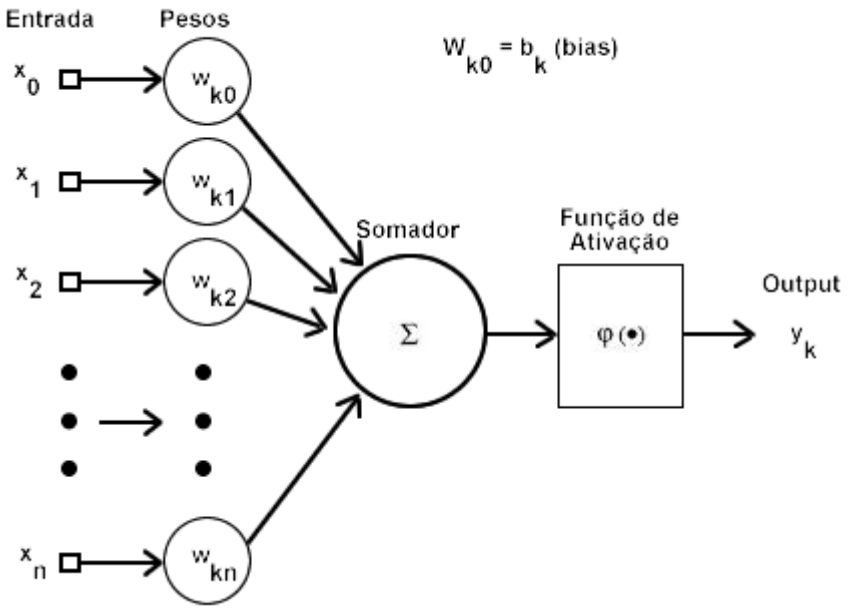


Figura 2.1: Representação do Neurônio Artificial

CONJUNTO DE SINAPSES

O conjunto de sinapses ou elos de conexão são as entradas do neurônio. Cada entrada possui uma força ou peso que representa o quanto a entrada é importante para o neurônio. Cada conexão de entrada pode ser excitatória ou inibitória para o neurônio. As excitatórias contribuirão para que o neurônio seja ativado e o fluxo de informação passe adiante e as inibitórias contribuirão para que o neurônio seja desativado e o fluxo não passe adiante.

BIAS

O bias ou viés é o peso sináptico atribuído ao próprio neurônio. Além de todas as entradas sinápticas terem pesos o próprio neurônio também o tem. Este peso afeta o neurônio da mesma forma que o peso de uma entrada sináptica, apesar de ser considerado sempre com valor 1 a entrada que será multiplicada pelo peso bias. Assim como na entrada sináptica o bias pode ser tanto inibitório quanto excitatório.

SOMADOR

Este constitui um combinador linear, não é nada mais do que a soma ponderada de cada entrada multiplicada pelo seu peso sináptico, e ao final é adicionado o bias. O resultado gerado pelo somador é o insumo para a função de ativação.

FUNÇÃO DE ATIVAÇÃO

A função de ativação ou função restritiva ou função de transferência recebe o resultado do somador e restringe a amplitude da saída do neurônio. Isto faz com que o neurônio receptor deste sinal não receba um sinal fora do escopo de amplitude de sinal proposta pela rede. Existem várias funções de ativação, as mais comuns são a listadas na Tabela 2.1.

Nome da função	Uso/Características	Função
Linear	Usada para buffers de entrada e saída de dados. É usada também em BSB e Hopfield.	$f(s) = s$
Sinal	Utilizada em paradigmas como Perceptron	$f(s) = \begin{cases} +1 & \Rightarrow s \geqslant 0 \\ -1 & \Rightarrow s < 0 \end{cases}$
Passo	Pode ser usada no lugar da função de transferência Perceptron, ou em BAM baseada em 0	$f(s) = \begin{cases} +1 & \Rightarrow s \geqslant 0 \\ 0 & \Rightarrow s < 0 \end{cases}$
Hopfield/BAM	Usadas em redes Hopfield e BAM. Caso s=0, mantém o valor de saída anterior	$f(s) = \begin{cases} +1 & \Rightarrow s > 0 \\ -1 & \Rightarrow s < 0 \\ inateraldo & \Rightarrow s = 0 \end{cases}$
BSB ou Limiar Lógico	Linear entre os limites inferior e superior, prendendo os valores de saída ao limite caso algum destes seja excedido. Usado em BSB	$f(s) = \begin{cases} -k & \Rightarrow s \leqslant -k \\ s & \Rightarrow -k < s < +k \\ +k & \Rightarrow s \geqslant +k \end{cases}$
Linear de Limites Rígidos	Linear entre os limites 0 e 1, prendendo os valores de saída ao limite caso algum destes seja excedido. Usado em paradigmas de competição através de inibição	$f(s) = \begin{cases} 0 & \Rightarrow s \leqslant 0 \\ s & \Rightarrow 0 < s < +1 \\ +1 & \Rightarrow s \geqslant +1 \end{cases}$
Logística	Função de transferência sigmoidal usada tradicionalmente em redes feedforward de aprendizagem backpropagation. Usada também em redes Hopfield, BAM e BSB	$f(s) = \dfrac{1}{1+e^{-s}}$
Hiperbólica	Função usada muitas vezes no lugar da função logística por variar de -1 a 1 e não de 0 a 1 como na função logística	$f(s) = \dfrac{1-e^{-2s}}{1+e^{-2s}}$
Gaussiana	Saída é simétrica em torno de um centro associado, utilizadas em redes RBF (redes de funções de base radial).	$f(s) = e^{\left[\dfrac{-(x-x_m)^2}{2\varphi}\right]}$

Tabela 2.1: Principais funções de ativação

2.2.2 MESO-ESTRUTURA

A Meso-Estrutura define como será a arquitetura da rede, ou seja, define a organização e o arranjo físico dos neurônios, como na representação da Figura 2.2. Nela são consideradas a quantidade de camadas da rede, a quantidade de neurônios por camada, os tipos de conexões e o grau de conectividade entre os neurônios.

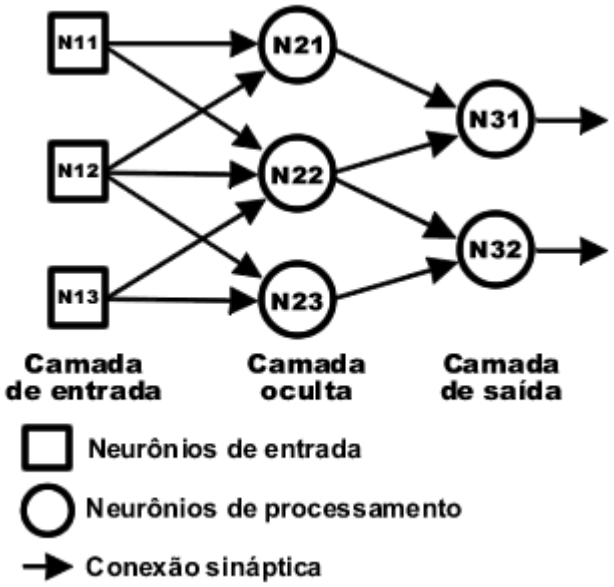


Figura 2.2: Representação de uma Rede Neural Artificial 3x3x2

QUANTIDADE DE CAMADAS

Este fator informa quantas camadas a arquitetura da rede possuirá. Normalmente redes têm uma camada de entrada, uma ou mais camadas ocultas, e uma camada de saída. Contudo, há redes como a de Hopfield que possuem apenas uma camada, onde a camada de entrada e a de saída são a mesma.

QUANTIDADE DE NEURÔNIOS POR CAMADA

Este fator informa quantos neurônios serão utilizados em cada camada da rede neural. Entre um neurônio e outro se encontram as conexões, então para que uma rede aprenda devemos ter uma quantidade de neurônios satisfatória. A quantidade de neurônios nas camadas de entrada e saída são definidas pela sua aplicação, mas a quantidade de neurônios na camada oculta é uma incógnita, uma quantidade pequena de neurônios faz com que a rede não consiga aprender tudo

e uma quantidade grande de neurônios faz com que a rede demore a aprender, decore padrões, não generalize.

TIPOS DE CONEXÕES

Os tipos de conexões informam como serão as características das sinapses na rede neural, ou seja, se serão do tipo forward (propagação), backward (retropropagação) ou lateral. Forward é a ligação sináptica que tem sentido da camada anterior para a posterior. Backward é a ligação sináptica que tem sentido da camada posterior para a anterior. Lateral é a ligação sináptica que tem sentido para os neurônios de mesma camada.

GRAU DE CONECTIVIDADE

O grau de conectividade é um fator comumente usado para definir a forma de ligação entre os neurônios. Normalmente o grau de conectividade se aplica a todos os neurônios da rede. Alguns graus de conectividade são Um a Um, Plena, Esparsa e Randômica.

- Um a Um - Cada neurônio de uma camada somente está conectado a um neurônio da camada posterior.
- Plena - Todos os neurônios de uma camada estão conectados aos neurônios da camada posterior.
- Esparsa - As ligações entre os neurônios de uma camada e os da camada posterior são seletivas, isto significa que nem todos os neurônios de uma camada é ligado a todos os neurônio da camada posterior, normalmente utiliza-se este tipo de ligação para otimizar a aprendizagem.
- Randômica - As ligações entre os neurônicos de uma camada aos neurônios da camada posterior são aleatórias.

2.2.3 MACRO-ESTRUTURA

A Macro-Estrutura define como redes neurais podem trabalhar em conjunto, os chamados sistemas de redes. Existem duas formas de redes neurais trabalharem em conjunto. A primeira é acoplar redes dentro de uma nova estrutura, formando assim uma nova rede, chamada rede híbrida e também chamada redes fortemente acopladas. A segunda forma é conectar as redes sem que haja alteração de suas estruturas. Ao projetar um sistema de redes neurais a primeira

dificuldade encontrada é o não entendimento do problema da aplicação de forma modular, isto é, não se sabe como dividir o problema em partes pequenas para que se possa criar redes de finalidades específicas. As maiores dificuldades na modelagem de um sistema de redes é especificar a quantidade, o tipo, o tamanho das redes de computadores, e especificar seus padrões de conexões, isto por que é difícil mapear o passo a passo das atividades que a rede fez para conseguir chegar a determinado resultado.

2.3 DINÂMICA

A dinâmica de uma rede é a forma com que ela processa seus dados. Esta forma de processamento está relacionada diretamente com a estrutura adotada na construção da rede. Há basicamente duas dinâmicas para as redes neurais: Sem ciclos e com ciclos.

2.3.1 ACÍCLICA

São redes onde o fluxo da informação seguem sempre adiante, não há realimentação dos neurônios, isto significa que o fluxo de informação passa somente uma vez por neurônio. Redes com essas características são chamadas de redes Feedforward (ex: perceptron multi-camadas).

2.3.2 CÍCLICA

São redes onde o fluxo da informação passa pela rede e é reutilizado como uma das entradas da rede, camada ou um neurônio em particular, os neurônios são realimentados por neurônios adiante no fluxo, fazendo com que o fluxo de informação passe mais de uma vez pelo neurônio. Redes com estas características são chamadas de redes 'Recorrentes' (ex: Hopfield). Em redes recorrentes a finalização do fluxo se dá quando, após as realimentações, as saídas permanecem inalteradas, então a rede se tornou estável. Contudo, isso não quer dizer que a rede aprendeu de forma correta. Existem vários estados estáveis para uma rede, mas precisamos do melhor, por ser o mais confiável para que a rede responda o mais preciso possível. Precisamos garantir que a rede convergirá para uma solução quando dado um padrão de entrada. Então temos um problema que é a estabilidade da rede. Nesse ponto entra o Teorema de Lyapunov, ele descreve que se podemos definir uma função de energia para uma rede, e se esta função atende a certos critérios, então o sistema será estável globalmente. Uma forma de se conseguir o melhor estado de estabilidade é a utilização de Boltzmann Machine, que emprega métodos estatísticos. Outra forma seria a utilização de Algoritmos Genéticos para a procura do melhor estado estável,

utilizando-se de evolução e seleção natural.

2.4 APRENDIZAGEM

O processo de aprendizagem pode ser definido de forma sintética como a aquisição de novos conhecimentos, desenvolvimento de competências e mudança de comportamentos. Para isso, o campo de redes neurais artificiais utiliza-se da Tecnologia Behaviorista, que é o estudo do comportamento, e das Aproximações Cognitivas, que é o estudo dos fenômenos da mente como percepção, memória, aprendizagem e linguagem enquanto processos relativos ao percurso mental da informação. O comportamento de uma rede neural artificial é alterado para que ela trabalhe de acordo com uma finalidade e para isso precisamos fazer com que ela aprenda um conjunto de funções necessárias para alcançar esse objetivo. Essa aprendizagem se baseia nas idéias de condicionamento, e são aplicadas em uma rede neural artificial através da Metodologia de Aprendizagem, Atribuição de Crédito e Algoritmo de Aprendizagem.

2.4.1 CONDICIONAMENTO

Condicionamento é o processo de associação de uma reação reflexa a um estímulo que habitualmente não a desencadeia, este conceito é importante para o entendimento da lógica dos métodos de ensinamento.

CONDICIONAMENTO CLÁSSICO

Neste condicionamento é apresentado um par de estímulos. Um estímulo que já é conhecido e um que é estranho, após apresentar diversas vezes esse par de estímulos, o objeto a ser condicionado passa a responder ao estímulo estranho da mesma forma que ao estímulo conhecido. Assim formam-se novas conexões sinápticas inicialmente temporárias e conforme a repetição de apresentação do par de estímulos, as conexões passam a ser duradouras. Suponha um objeto a ser condicionado *oc* que quando lhe é apresentado o estímulo e_1 ele responde r_1, e quando lhe é apresentado e_2, ou *oc* não reposnde ou responde de forma não interessante ao estudo. Para efetuarmos o condicionamento apresentamos um estímulo e_2 seguido do estímulo e_1 para que *oc* responda r_1, assim, após algumas apresentações *oc* passará a responder r_1 ao estímulo e_2.

CONDICIONAMENTO OPERANTE

Este condicionamento trabalha sobre a frequência de uma reposta dado um estímulo. Isso quer dizer que há uma modelagem da resposta do objeto a ser condicionado através de reforço, punição e aproximações sucessivas. Para cada estímulo apresentado ao objeto a ser condicionado haverá uma resposta produzida por ele, esta resposta gera uma consequência que afeta a probabilidade da resposta ocorrer novamente apresentando o mesmo estímulo. A consequência pode ser punitiva, o que reduzirá a frequência, ou reforçadora, o que aumentará a frequência. No aprendizado operante ou aprendizado instrumental, há o envolvimento de três componentes:

- Feedback - Este componente diz que para cada ação solicitada existe uma reação desejada, caso a reação seja positiva há uma bonificação, caso a reação seja negativa há uma punição;
- Treinamento baseado em respostas previamente conhecidas - Este componente diz que dada uma ação, a reação dependerá dos limites comportamentais do objeto a ser condicionado, isso quer dizer que ao treinarmos devemos começar pelos comportamentos conhecidos;
- Treinamento progressivo - Este componente diz que um objeto a ser condicionado tem um desempenho mais eficiente quando se utiliza amostragem. O condicionamento ocorre por demonstrações, assim o objeto a ser condicionado aprende mais rápido e aprende comportamentos mais complexos.

2.4.2 METODOLOGIA DE APRENDIZAGEM

A metodologia de aprendizagem define como a rede aprenderá, se é com um professor onde há correções sobre as ações da rede, se é com um crítico que dará a direção mais provável com base nas ações anteriores da rede ou por observação ao ambiente onde a rede consegue classificar padrões de elementos no ambiente.

APRENDIZAGEM SUPERVISIONADA

Neste tipo de aprendizagem a rede é treinada por um professor. Esse professor tem total conhecimento do ambiente, isto quer dizer que, ele conhece aquele assunto ao qual a rede será treinada, para cada situação que o ambiente apresenta o professor tem uma reação correta.

No treinamento da rede simulamos situações do ambiente para a rede e para cada simulação a rede responderá de alguma forma. Ao simularmos para o professor a mesma situação apresentada a rede, ele sempre dará a resposta correta, pois conhece do assunto. Então comparamos a resposta da rede e a do professor e aplicamos um corretivo à rede com algum algoritmo de aprendizagem. Este algoritmo irá fazer correção dos pesos sinápticos dos neurônios. Esse processo de simulação do ambiente, comparação das respostas rede/professor e correção dos pesos sinápticos se dá várias vezes até que a rede emule as ações do professor e por conseguinte aprenda a responder corretamente por conta própria.

APRENDIZAGEM NÃO SUPERVISIONADA

Nesta aprendizagem, não há um professor externo ou crítico para supervisionar o processo de aprendizagem. Para a rede são dadas condições para que ela possa qualificar seus resultados dada a cada situação do ambiente. Com isso ela tem a liberdade de criar suas próprias classificações das situações do ambiente.

Essas condições são dadas graças aos algoritmos para aprendizagem auto-organizada. Esses algoritmos têm como objetivo descobrir padrões significativos ou características nos dados de entrada. Eles dispõem de um conjunto de regras de natureza local (a modificação aplicada ao peso sináptico de um neurônio depende de sua vizinhança imediata), que o capacitam a aprender a calcular um mapeamento de entrada-saída com propriedades específicas desejáveis.

Redes com finalidade de auto-organização tendem a seguir estruturas neurobiológicas de uma maneira muito mais extensa do que uma para finalidade de aprendizagem supervisionada. Esse tipo de aprendizagem pode ser muito útil para redes com finalidade de classificação de informação em um Data Warehouse, que é um sistema de computação utilizado para armazenar, em bancos de dados e de forma consolidada, as informações relativas às atividades de uma organização, essa classificação é útil para fins de estatística.

APRENDIZAGEM POR REFORÇO

Nesta aprendizagem, o aprendizado de um mapeamento de entrada-saída é realizado através de interação contínua com o ambiente, visando minimizar um índice escalar de desempenho. No aprendizado supervisionado existia o papel do professor que informava a resposta certa para cada resposta da rede. No caso do aprendizado por reforço nós temos o crítico, ele é responsável por orientar a rede informando-a se ela está indo bem ou mal em suas respostas com base em respostas anteriores e cada reforço primário baseado na situação do ambiente.

O crítico observa uma seqüência temporal de estímulos (entradas) que foi recebida pela rede e gera um sinal de reforço heurístico. Seu objetivo é minimizar uma função de custo para avançar, que é o custo cumulativo das ações tomadas anteriormente. Ele faz isso justamente para o caso de haver uma tomada de decisão anterior naquela seqüência que seja melhor determinante do comportamento global do sistema.

2.4.3 ATRIBUIÇÃO DE CRÉDITOS

A atribuição de créditos determina que a rede tem seus parâmetros livres alterados com bonificação ou punição para que haja a aprendizagem. Essa atribuição está ligada ao resultado global da rede. Se a rede responde de forma correta há ligações que devem ser reforçadas, e de forma contrária, se ela responde de forma errada há ligações que devem ser punidas. Os algoritmos de aprendizagem servem para alterar os parâmetros livres.

2.4.4 ALGORITMO DE APRENDIZAGEM

O algoritmo de aprendizagem define quais ligações sinápticas serão bonificadas ou punidas e o quanto o seu peso será alterado, aumentando-o ou diminuindo-o. Cada algoritmo tem sua forma de alterá-los para adequar-se melhor à estrutura e dinâmica da rede. Esta forma determina a maneira como ocorre a modificação dos parâmetros livres.

POR CORREÇÃO DE ERRO

A aprendizagem por correção de erro se dá através das correções dos pesos sinápticos a partir de um algoritmo de aprendizagem. Para efetuar as operações com o algoritmo de aprendizagem temos um processo bem simples (apesar de os algoritmos de aprendizagem em si serem relativamente complexos pela utilização de derivadas e gradientes).

Dada uma entrada para a rede neural, ela reagirá com uma resposta. Esta resposta de saída da rede é comparada com a resposta desejada. A diferença entre as respostas será o sinal de erro. Neste ponto o erro aciona um mecanismo de controle. Este mecanismo é responsável pelos ajustes corretivos feitos nas conexões sinápticas da rede. A cada iteração desta rede em seu processo de aprendizagem são feitos os ajustes dos pesos. Esses ajustes podem ser obtidos minimizando-se uma função de custo ou índice de desempenho. Para ajustar o peso de cada conexão sináptica nós multiplicamos o sinal de erro pelo valor de entrada da conexão sináptica. Deste resultado multiplicamos por uma constante positiva definida como taxa de aprendizagem. Este novo resultado é somado ao valor do peso da conexão. Esta conexão passa a ter este novo

valor como forma de seu aprendizado. Este processo é aplicado em todas as conexões sinápticas da rede. Os ajustes continuam até que a rede se torne estável. O excesso de treinamento da rede pode acarretar em um super-ajustamento ao conjunto de treinamento e conseqüentemente fazer com que, em certo ponto, o desempenho caia em relação ao conjunto de teste, com isso a rede passará a responder aos estímulos de forma errada.

Este processo de aprendizagem é denominado aprendizagem por correção de erro e a minimização da função de custo resulta na regra Delta ou regra Widrow-Hoff (nome em homenagem a seus criadores Bernard Widrow e Marcian Hoff). A regra Delta pode ser formulada como: O ajuste feito em um peso sináptico de um neurônio é proporcional ao produto do sinal de erro pelo sinal de entrada da sinapse em questão. Nota-se que nesta aprendizagem a resposta desejada é obtida a partir de uma fonte externa à rede. Na prática podemos exemplificar uma iteração da seguinte forma: Considere um neurônio k como o único da camada de saída de uma rede neural alimentada adiante. O neurônio k é acionado por dois neurônios da camada de entrada, estes são representados por elementos dentro de um vetor de sinal de entrada X como x_1 e x_2 e sua ligação com o neurônio k tem os pesos w_{k1} e w_{k2} respectivamente. Ao aplicar o vetor X na rede, esta processará e dará uma resposta $y_k = \sum_{i=0} x_i.w_{ki}$. Esta resposta será subtraída da resposta desejada d_k gerando assim um erro $e_k = d_k - y_k$. Para fazermos os ajustes definimos uma taxa de aprendizado n que é uma constante positiva. Os pesos são atualizados da seguinte forma:

$$w_{kj} = w_{kj}^{z^{-1}} + \{n.e^{z^{-1}}.x_k^{z^{-1}}\},$$

onde z^{-1} representa um passo de tempo anterior, ou seja um atraso.

APRENDIZAGEM BASEADA EM MEMÓRIA

Como o próprio nome diz, nesta aprendizagem a classificação se dá a partir de experiências passadas armazenadas em uma grande memória de exemplos de entrada-saída classificados corretamente. Isto significa que esta grande memória é um conjunto e cada elemento deste conjunto é um par de entrada e saída desejada, e podemos escrever sua forma matemática da seguinte maneira:

$$\{(x_i, d_i)\}_{i=1}^{N},$$

onde:

- x_i representa o vetor de entrada;
- d_i representa a resposta desejável, sendo um escalar ou não;
- i representa o índice de elementos;
- N representa o número de elementos do conjunto.

Para a classificação de um vetor de teste x_{teste}, o algoritmo de aprendizagem baseado em memória responde buscando e analisando os dados de treinamento em uma vizinhança local ao vetor de teste x_{teste}. Esta classificação envolve dois itens essenciais e estes itens farão os algoritmos para este modelo de aprendizagem serem diferentes entre si. Estes dois itens são:

- O critério utilizado para definir a vizinhança local à x_{teste};
- A regra de aprendizagem aplicada aos exemplos de treinamento na vizinhança local a x_{teste}.

Regra do vizinho mais próximo - A forma mais simples, mas efetiva, de aprendizagem baseada em memória é conhecida como a regra do vizinho mais próximo. Nesta regra a vizinhança local é definida como o exemplo de treinamento que se encontra na vizinhança imediata do vetor de teste x_{teste}.

Diz-se que

$$x'_N \in \{x_1, x_2, ..., x_N\}$$

é o vizinho mais próximo de x_{teste} se

$$mind(x_i, x_{teste}) = d(x'_N, x_{teste}),$$

onde:

- $d(x'_N, x_{teste})$ é a distancia euclidiana entre os vetores x_i e x_{teste};
- x'_N é a classificação de x_{teste}.

Esta regra é independente da distribuição fundamental responsável pela geração dos exemplos de treinamento. Covert e Hart em 1967 (MEAD, 1996b) estudaram formalmente esta regra como uma ferramenta para classificação de padrões. Sua análise baseia-se em duas proposições:

- Os exemplos classificados (x_i, d_i) são independentemente e identicamente distribuídos (iid), de acordo com a distribuição de probabilidade conjunta do exemplo (x, d);
- O tamanho da amostra N é infinitamente grande.

Considerando estas duas proposições, temos que a probabilidade de erro de classificação é limitada acima pelo dobro da probabilidade de erro bayesiana, isto é, a mínima probabilidade de erro entre todas as regras de decisão. O surpreendente desta regra é que o vizinho mais próximo contém metade da informação sobre a classificação de um conjunto de treinamento de tamanho infinito.

Regra dos k vizinhos mais próximos - Esta regra é uma variação da regra do vizinho mais próximo, e tem como objetivo fazer classificação, não pela classe do vizinho mais próximo, mas sim por uma votação democrática das classes dos k vizinhos mais próximos. Encontram-se os k vizinhos mais próximos, a classe de x_{teste} será a classe da maioria dos vizinhos.

APRENDIZAGEM HEBBIANA

A mais antiga e mais famosa de todas as regras de aprendizagem é "O postulado de aprendizado de Hebb", que tem essa denominação em homenagem ao neuropsicólogo Donald Olding Hebb.

"Quando um axônio da célula *A* está perto o suficiente para excitar uma célula *B* e participa do seu disparo repetida ou persistentemente, então algum processo de crescimento ou modificação metabólica acontece em uma das células ou em ambas, de tal forma que a eficiência de *A* como uma das células que dispara *B* é aumentada". (HEBB, 1949b)

Hebb propôs que uma base da aprendizagem associativa (no nível celular) fosse esta modificação metabólica que fazia aumentar a eficiência da comunicação entre as células, e que resultaria em uma modificação permanente do padrão de atividade de um "agrupamento de células nervosas" espacialmente distribuídos. Tal afirmação tem seu contexto neurobiológico, mas podemos expandi-la e reescrevê-la em uma regra de duas partes.

Se dois neurônios em ambos os lados de uma sinapse são ativados sincronamente, então a força daquela sinapse é seletivamente aumentada;

Se dois neurônios em ambos os lados de uma sinapse são ativados assincronamente, então aquela sinapse é seletivamente enfraquecida ou eliminada (STENT, 1973) e (CHANGEUX et al., 1976).

Tal sinapse é denominada sinapse hebbiana, e é definida como uma sinapse que usa um

mecanismo dependente do tempo, altamente local e fortemente interativo para aumentar a eficiência sináptica como uma função da correlação entre as atividades pré-sinápticas e pós-sinápticas. Sendo assim, deduz-se os quatro mecanismos (propriedades) fundamentais que caracterizam uma sinapse hebbiana:

Mecanismo dependente do tempo - Este mecanismo refere-se ao fato de que as modificações em uma sinapse hebbiana dependem do tempo exato de ocorrência dos sinais pré-sinápticos e pós-sinápticos.

Mecanismo local - Pela sua natureza, uma sinapse é um local de transmissão onde sinais portadores de informação (representando a atividade incidente nas unidades pré-sináptica e pós-sináptica) estão em contigüidade espaço-temporal. Esta informação localmente disponível é utilizada por uma sinapse hebbiana para produzir uma modificação sináptica local que é específica para a entrada.

Mecanismo interativo - A ocorrência de uma modificação em uma sinapse hebbiana depende dos sinais em ambos os lados da sinapse. Isto é, uma forma de aprendizagem hebbiana dependente de uma 'interação verdadeira' entre os sinais pré-sináptico e pós-sináptico, no sentido de que não podemos fazer uma previsão a partir de apenas uma dessas duas atividades. Note também que esta dependência ou interação pode ser de natureza determinística ou estatística.

Mecanismo conjuncional ou correlativo - Uma interpretação do postulado de aprendizado de Hebb é que a condição para uma modificação da eficiência sináptica é a conjunção dos sinais pré-sináptico e pós-sináptico. Assim, de acordo com esta interpretação, a ocorrência simultânea dos sinais pré-sináptico e pós-sináptico (dentro de um curto intervalo de tempo) é suficiente para produzir a modificação sináptica. É por esta razão que uma sinapse hebbiana é algumas vezes denominada sinapse conjuncional. Para uma outra interpretação do postulado de aprendizado de Hebb, podemos considerar o mecanismo interativo que caracteriza uma sinapse hebbiana em termos estatísticos. Em particular, a correlação temporal entre os sinais pré-sinaptico e pós-sináptico é vista como sendo responsável por uma modificação sináptica. Neste sentido, uma sinapse hebbiana é também denominada uma sinapse correlativa. A correlação é de fato a base da aprendizado (EGGERMONT, 1990).

Reforço e Depressão Sinápticos - Uma sinapse hebbiana possui processos para se reforçar, aumentando assim sua eficiência de comunicação entre o par de neurônios, quando reconhece uma atividade positivamente correlacionada, contudo, não inclui processos adicionais que podem resultar no enfraquecimento da sinapse ao reconhecer uma atividade negativamente correlacionada ou não-correlacionada. A depressão sináptica também pode ser do tipo não interativo. Especificamente, a condição do tipo interativo pode levar em consideração, simplesmente,

a atividade não-coincidente pré-sináptica ou pós-sináptica. Podemos classificar as sinapses em três grupos:

- Hebbiana - Aumenta sua força com sinais pré-sináptico e pós-sináptico positivamente correlacionados e diminui sua força com sinais pré-sináptico e pós-sináptico negativamente correlacionados ou não-correlacionados.
- Anti-Hebbiana - Esta sinapse trabalha de forma inversa à Hebbiana, diminui sua força com sinais pré-sináptico e pós-sináptico positivamente correlacionados e aumenta sua força com sinais pré-sináptico e pós-sináptico negativamente correlacionados ou não-correlacionados.
- Não-Hebbiana - Esta sinapse trabalha de tal forma que seu mecanismo funcional não tem qualquer relação com o mecanismo das sinapses hebbianas em termos de correlações de aprendizagem.

Tanto em uma sinapse hebbiana quanto em uma sinapse anti-hebbiana, podemos ver que a modificação de sua eficiência está diretamente dependente do tempo, é altamente local e sua natureza é fortemente interativa. Neste sentido podemos dizer que uma sinapse anti-hebbiana é de natureza hebbiana, apesar de não o ser funcionalmente.

Modelos matemáticos de modificações hebbianas - Para formular a aprendizagem hebbiana em termos matemáticos, consideremos um peso sináptico w_{kj} pertencente ao neurônio k com sinais pré-sinápticos e pós-sinápticos representados por x_j e y_k, respectivamente. O ajuste aplicado ao peso sináptico w_{kj} no passo de tempo n é expresso na forma geral:

$$\Delta w_{kj} = F(y_k(n), x_j(n)),$$

onde $F()$ é uma função tanto do sinal pré-sináptico quanto do sinal pós-sináptico. Normalmente os sinais x_j e y_k são tratados como adimensionais.

Hipótese de Hebb - Esta é a forma mais simples de aprendizagem e sua forma matemática é a seguinte:

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n),$$

onde η representa uma constante positiva que determina a taxa de aprendizagem.

Esta equação deixa bem visível a natureza correlativa de uma sinapse hebbiana. Esta

equação é referida, às vezes, como a regra do produto das atividades.

Hipótese de covariância - Esta hipótese é baseada na hipótese de Hebb, a diferença é que os sinais pré-sináptico e pós-sináptico são substituídos pelo desvios dos sinais pré-sináptico e pós-sináptico em relação aos seus respectivos valores médios em um certo intervalo de tempo. Sua forma matemática é a seguinte:

$$\Delta w_{kj}(n) = \eta(x_j - \bar{x})(y_k - \bar{y}),$$

onde η representa uma constante positiva que determina a taxa de aprendizagem, $\bar{x}$ e $\bar{y}$ representam os valores médios no tempo dos sinais pré-sinápticos x_j e pós-sináptico y_k respectivamente.

A hipótese de covariância nos permite chegar a uma convergência não-trivial em $x_k = \bar{x}$ e $y_j = \bar{y}$ fornecendo uma previsão da potencialização e da depressão sináptica, ela pode ser facilmente entendida se a imaginarmos como um AND lógico, pois o peso sináptico w_{kj} somente é reforçado quando os sinais pré-sináptico e pós-sináptico são maiores que seus valores médios caso contrário há uma depressão sináptica.

APRENDIZAGEM COMPETITIVA

A aprendizagem competitiva tem este nome pelo fato de que todos os neurônios de saída estão disputando quem se tornará ativo. Nela apenas um neurônio de saída pode estar ativo em determinado instante. Esta característica torna esta forma de aprendizagem muito adequada para descobrir características estaticamente salientes que podem ser utilizadas para classificar um conjunto de padrões de entrada. Na regra de aprendizagem competitiva existem três elementos básicos (HAYKIN, 2001):

- O primeiro elemento é o conjunto de neurônios, onde a única diferença entre cada neurônio são alguns pesos sinápticos distribuídos aleatoriamente, fazendo com que para cada conjunto de padrões de entrada haja uma resposta diferente;
- O segundo elemento é o limite imposto sobre a força de cada neurônio;
- O terceiro e último elemento é um mecanismo que permite que o neurônio compita pelo direito de ser o único a responder pela rede (ou por um grupo de neurônios) em determinado instante. Existe uma denominação para o único neurônio responsável pela resposta e vencedor da competição, chamamos de Neurônio Vencedor Leva Tudo.

Desta forma os neurônios competidores passam a se especializar em agrupamentos de padrões de entrada similares, tornando-se assim, detectores de características para classes diferentes de padrões de entrada. A forma de saber qual é o Neurônio Vencedor Leva Tudo a partir de um conjunto de entradas é comparando o campo local induzido de cada neurônio. O neurônio com maior campo local induzido será o vencedor. O campo local induzido é gerado por cada neurônio da última camada, e ele é uma combinação das ligações que estes neurônios tem com os neurônios da camada anterior. Uma forma de tentar dar mais distinção à essa competição é fazer ligações inibitórias entre os neurônios da última camada, ligando todos a todos. O sinal de saída do neurônio vencedor passa a ser 1 (um) enquanto o sinal de saída dos demais neurônios da última camada passam a ser 0 (zero).

$$y_k = \begin{cases} 1 & \Rightarrow v_k \, v_j (\forall j, j \neq k) \\ 0 & \Rightarrow v_k \leq v_j \end{cases},$$

onde:

- y_k sinal de saída do neurônio k para um padrão de entrada;
- v_k campo local induzido do neurônio k para um padrão de entrada que é a combinação de todas as entradas diretas e as realimentadas;
- v_j campo local induzido do neurônio j. Varre os neurônios em busca de cada campo local induzido para compará-los com o campo local induzido do neurônio k.

Considerando w_{kj} o peso sináptico conectando o neurônio de entrada j ao neurônio k, suponha que a cada neurônio seja alocada uma quantidade fixa de peso sináptico que é positiva e distribuída entre seus nós de entrada, então teremos:

$$\sum_j w_{kj} = 1 (\forall k),$$

O aprendizado se dá através do deslocamento dos pesos sinápticos das conexões inativas para as conexões ativas. De acordo com a regra de aprendizagem competitiva temos a variação Δw_{kj} aplicada ao peso $w + kj$ como segue:

$$\Delta w_{kj} = \begin{cases} \eta (x_j - w_{kj}) & \Rightarrow \textit{se o neuronio k vencer a rodada} \\ 0 & \Rightarrow \textit{se o neuronio k perder a rodada} \end{cases},$$

onde η é a taxa de aprendizagem.

Esta regra tem o efeito global de mover o vetor de peso sináptico w_k do neurônio vencedor k em direção ao padrão de entrada x.

APRENDIZAGEM DE BOLTZMANN

Seu nome foi dado em homenagem a Ludwig Boltzmann. Este algoritmo de aprendizagem estocástico deriva de idéias enraizadas na mecânica estatística. A aprendizagem de Boltzmann é denominada uma Máquina de Boltzmann. Os neurônios constituem uma estrutura recorrente de operação binária: quando um neurônio está em um estado ativo é representado por $+1$, quando está num estado inativo é representado por -1. A máquina é caracterizada por uma função de energia, e cujo valor é determinado pelos estados particulares ocupados pelos neurônios individuais da máquina:

$$E = \frac{-1}{2}\sum_{j}\sum_{k} w_{kj}x_k x_j, j \neq k,$$

onde:

- x_j é o estado do neurônio j;
- w_{kj} é o peso sináptico conectando o neurônio j ao neurônio k.

A especificação de $j \neq k$ simboliza a ausência de auto-realimentação na Máquina de Boltzmann. A máquina opera escolhendo aleatoriamente um neurônio k em um determinado passo do processo de aprendizagem trocando o seu estado de x_k para $-x_k$ a uma temperatura T com probabilidade.

$$P(x_k \rightarrow -x_k) = \frac{1}{1+\exp(-\frac{\Delta E_k}{T})},$$

onde ΔE_k é a variação de energia (variação da função de energia da máquina) resultante daquela troca.

Como não estamos trabalhando com mecânica estatística mas apenas pegando emprestado seus conceitos considerem T como uma pseudo-temperatura. Aplicando esta regra repetidamente a máquina achará o equilíbrio térmico. Neste tipo de aprendizagem os neurônios dividem-se em dois grupos: visíveis e ocultos; O neurônios visíveis fazem a ligação entre a

rede e o ambiente, enquanto os neurônios ocultos operam livremente. São considerados dois modos de operação:

- Condição Presa - Onde cada neurônio visível está preso a um estado específico determinado pelo ambiente;
- Condição de Operação Livre - Onde todos os neurônios da rede, ocultos ou visíveis, podem operar livremente.

Considerando ρ_{kj}^{+} a correlação entre os estados dos neurônios j e k, com a rede na condição presa e, ρ_{kj}^{-} a correlação entre os estados dos neurônios j e k, com a rede na condição de operação livre. Ambas correspondem às médias sobre todos os estados possíveis da máquina, quando ela está em equilíbrio térmico. Então o ajuste feito no peso sináptico w_{kj} do neurônio j para o neurônio k é a variação Δw_{kj}.

$$\Delta w_{kj} = \eta(\rho_{kj}^{+} - \rho_{kj}^{-}), j \neq k,$$

onde η é a taxa de aprendizagem.

Tanto ρ_{kj}^{+} quanto ρ_{kj}^{-} assumem valores entre -1 e 1.

2.5 IMPLEMENTAÇÕES

2.5.1 IMPLEMENTAÇÃO EM SOFTWARE

A maioria das implementações de redes neurais é feita em software. Implementações em software são as de custo mais baixo e as de maior flexibilidade, por isso no desenvolvimento de um projeto de redes neurais, elas são as primeiras a serem construídas. Nelas são feitos os testes de potencial entre um leque de redes que se pré-dispõe a solucionar um determinado modelo de problema. Implementações em software podem ter sua finalidade para análise e aplicação de redes neurais, ou podem ser voltadas ao desenvolvimento de simuladores comerciais ou científicos. Simuladores para estes fins são construídos em linguagem de alto nível, e incorporam mais de um tipo de redes neurais. Geralmente eles são mais lentos do que aplicações desenvolvidas especificamente para um modelo. A utilização de redes neurais para a simulação deve-se a facilidade de implementação de modelos, verificação de correções da implementação, e controle de experimentos. A velocidade de operação da simulação fica atrelada à limitação do

poder computacional do hardware onde a simulação é executada e à ligação física do hardware, traçando assim a desvantagem deste tipo de implementação.

2.5.2 IMPLEMENTAÇÃO EM HARDWARE

Com certeza, a maior vantagem de se construir redes neurais artificiais em hardware é o alto desempenho para solucionar problemas de grande volume de processamento, e aplicabilidade para softwares em tempo real. Devido à sua natureza maciçamente paralela, uma rede neural se faz potencialmente rápida na computação em larga escala. Esta característica é completamente aplicável à tecnologia VLSI (Very-large-scale-integration) que fornece um meio de capturar comportamentos realmente complexos de uma forma altamente hierárquica (MEAD, 1996a). As implementações em hardware podem ser diferenciadas de acordo com as seguintes variáveis:

Escala de Integração - Esta variante informa qual é o nível de implementação, se é em nível de placa ou é em nível de chip. As implementações em nível de placas são as mais experimentais, enquanto as em nível de chips são as finais, justamente por ser mais caro produzir chips do que placas.

Tecnologia - É dada pelo comprimento do canal dos transistores. Quanto menor o comprimento dos transistores, maior será a quantidade transistores que será possível se ter numa mesma pastilha, e, conseqüentemente, será possível implementar mais neurônios artificiais e conexões sinápticas.

Densidade de Integração - A Densidade de Integração define quais operações serão realizadas pelo modelo da rede. Existem duas formas de integração que medem a sua densidade, a parcial e a completa. A completa parte do pressuposto de que a rede já foi treinada e testada, precisando apenas ser implementada em hardware para ser utilizada. A parcial parte do pressuposto que a rede será dinâmica podendo alterar seus pesos conforme o ambiente lhe impuser. Essa alteração de pesos é conseguida através do auxílio de placas aceleradoras e suas operações de ponto flutuante.

Modo de Integração - O Modo de Integração define formato do sinal que os neurônios artificiais adotarão para trabalhar na rede. Existem dois formatos de sinais, o digital e o analógico. A rede se dirá digital ou analógica de acordo com o funcionamento dos seus neurônios. O formato digital é o mais usado e mais fácil de ser implementado, por ser barato e exigir a tecnologia que os computadores suportam naturalmente, e, normalmente são desenvolvidos em cima da tecnologia CMOS (Complementary Metal-Oxide-Semiconductor). O analógico geralmente é desenvolvido utilizando EEPROM-MNOS, Floating Gate, e derivações de CMOS. Várias abor-

dagens diferentes têm sido utilizadas para se implementar redes neurais de forma analógica.

Precisão das Conexões - A Precisão das conexões define o número de bits utilizados para a representação dos pesos das conexões sinápticas. Os modelos ao serem treinados exigem uma maior precisão do que ao serem utilizados. A natureza dos modelos de redes faz com que algumas redes exijam maior precisão na representação do que outras.

Um grande problema encontrado na construção de redes neurais artificiais em hardware é a questão do espaço físico necessário para fazer as conexões sinápticas, que surge da necessidade de conectar centenas de pontos de entradas a um único neurônio, denominado de problema de Fan-in/Fan-out. Um agravante refere-se ao modo de integração analógico que necessita de certo grau de precisão das conexões. Outro agravante refere-se ao problema de mapeamento das conexões em uma área de silício, otimizando a área utilizada. Projetar e produzir placas e chips tem alto custo e a sua flexibilidade para implementar diferentes tipos de neurônios, arquiteturas e algoritmos de aprendizagem é pequena.

2.5.3 IMPLEMENTAÇÃO EM MEIOS ÓTICOS

Redes neurais implementadas em Meios Óticos são aquelas em que o mecanismo de sinapse é realizado em um sistema holográfico. Essa sinapse se dá por que ao expor à luz um cristal fotorefratário ou uma superfície fotorefratora, os elétrons na área luminosa são transferidos para as áreas escuras, formando um padrão de cargas no cristal. Esse padrão de cargas é que armazena os pesos das sinapses. Redes Neurais implementadas neste formato surgiram como solução à falta de espaço existente nas implementações em hardware. Estudos na Área de Redes Neurais Artificiais apontam a saída para esta falta de espaço físico com a utilização de hologramas. Os hologramas seriam capazes de armazenar cerca de $10^{12} bits/cm^3$. O sistema nervoso tem aproximadamente um total de 10^{11} neurônios, e seus estados ativo e inativo podem ser representados em apenas $1cm^3$ de um holograma (LOESCH, 1996). Existem três pontos fortes para se implementar redes em meios ópticos. O primeiro é a velocidade de acesso. O segundo é a alta capacidade de gravação. E o terceiro é a quebra dos limites de implementação em silício. Mas também existem três pontos fracos. O primeiro é o estágio inicial da engenharia por detrás das implementações. O segundo é a dificuldade de mapear as operações da luz sobre os meios óticos. E o terceiro é a lentidão do processo de adaptação, para que um cristal refratário possa armazenar uma informação é necessário expô-lo durante um longo período de tempo e à uma baixa conectividade.

2.6 ALGORITMO GENÉTICO COMO UMA ALTERNATIVA DE TREINAMENTO

Esta seção baseia-se no trabalho "Computação Evolutiva: Uma Abordagem Pragmática" de Fernando J. Von Zuben da UniCamp (ZUBEN, 2000). Os Algoritmos Genéticos fazem parte de uma classe particular de Algoritmos Evolutivos, pois eles fazem uso de técnicas inspiradas na biologia evolutiva. As principais técnicas são:

- Hereditariedade - Transmissão de características genéticas de um indivíduo a seus descendentes;
- Mutação - Modificação genética causada para tentar sobreviver no ambiente;
- Seleção Natural - Sobrevivem os mais aptos a um dado ambiente;
- Recombinação - Geração de descendentes.

Os Algoritmos Genéticos geralmente são utilizados para solucionar problemas de busca e otimização. Dada uma população com diversos indivíduos que podem solucionar um problema (uns melhores que outros), classificaremos todos com uma função 'Objetivo'. Vamos supor que a avaliação em relação ao objetivo vá de 0 a 10: 0 o pior indivíduo para o objetivo e 10 o melhor. Se na nossa população existir um indivíduo com a avaliação 10, então o utilizaremos para o que queremos, mas se não tivermos cruzaremos os melhores indivíduos, classificaremos seus filhos e uniremos eles à população. Descartaremos os piores e iremos repetir as operações de cruzamento, de classificação e de seleção dos mais aptos. Cada repetição de cruzamento, classificação e seleção é uma geração. Todo indivíduo tem um código genético e nele está contido suas características, essas características são avaliadas pela função objetivo. Há um passo que podemos incluir em um ciclo de geração chamado mutação. Este passo consiste em um indivíduo ter suas características adaptadas por alguma influencia e não por ter herdado de seus pais, isso faz com que haja uma maior variabilidade (Variação biológica entre indivíduos da mesma espécie, devendo-se, principalmente, aos fatores ambientais e genéticos) genética na população. Então, uma geração é composta de cruzamento, mutação, classificação e seleção. Na procura por um indivíduo que resolva da melhor forma um problema deve-se ter uma forma de finalizar o processo das gerações, ou até alcançar uma nota de avaliação ou por número de gerações.

2.6.1 ALGORITMO

Como dito anteriormente, o algoritmo tem sua lógica simples e natural, além de ser definida em poucos passos, como pode ser visto abaixo:

- Escolha da população inicial;
- Avalie cada indivíduo da população para um objetivo específico;
- Repita as operações até que se atinja a condição de parada;
  - Selecione os indivíduos com melhores avaliações para serem os genitores;
  - Gere novos indivíduos através de cruzamento e mutação;
  - Avalie os indivíduos gerados;
  - Inclua os indivíduos gerados na população e retire os piores indíviduos da população até que a mesma tenha a quantidade de indíviduos original;
- Retorna o indivíduo com a melhor avaliação;

Ou em outras palavras como segue no Algoritmo 1.

inicio
  ler: $populacao = lista\ de\ indivíduos;$
  ler: $FuncaoObjetivo = função\ de\ avaliação\ de\ indivíduo;$
  repita
    $listaPais \leftarrow SelecaoPais(populacao, FuncaoObjetivo)$;
    $listaFilhos \leftarrow Reproducao(listaPais)$;
    $populacao \leftarrow SelecaoNatural(populao, listaFilhos, FuncaoObjetivo)$;
  até *condicao-parada*;
  Resultado: $MelhorIndividuo(populacao, FuncaoObjetivo)$;
fim

Algoritmo 1: Algoritmo Genético

Onde:

- populacao → é um conjunto finito de indivíduos;
- individuo → é o portador das características que solucionam um problema;
- FuncaoObjetivo → é a função que classifica cada indivíduo de acordo com sua capacidade de solucionar um determinado problema;

- SelecaoPais → é a função que seleciona os genitores da próxima geração;
- listaPais → é a lista com os pais para cruzamento;
- Reproducao → é a função que cruza os pais e muta os descendentes ao cruzamento;
- listaFilhos → é a lista com os filhos gerados;
- SelecaoNatural → é a função que inclui os filhos gerados na população e retira os piores indivíduos para que a população volte a ter a quantidade original de indivíduos;
- condicao-parada → é a condição usada para que o algoritmo possa ser encerrado. Existem diversas condições de parada: avaliação de indivíduo, número de gerações, tempo, não há mais alterações significativas na avaliação de novas gerações de indivíduos, inspeção manual ou combinações dos elementos acima.
- MelhorIndividuo → é a função que retorna o melhor indivíduo em uma população dada uma função objetivo.

2.6.2 UTILIZAÇÃO COM REDES NEURAIS ARTIFICIAIS

Quando combinamos Algoritmos Genéticos com Redes Neurais Artificiais, a lógica como os Algoritmos Genéticos irão trabalhar é a mesma. A população continua formada por indivíduos, a geração continua tendo todas as suas operações, e continuamos a ter nossa função de avaliação como antes. Agora cada indivíduo é uma Rede Neural artificial, então precisamos entender como representar as redes como cromossomos para que possamos fazer as operações genéticas. Existem duas formas de representar as redes, a direta e a indireta, e a evolução da população é feita baseando-se na propriedade chamada de Informação Contextual, isso por que o conhecimento da rede está distribuido na sua estrutura, e o conhecimento fará com que a rede, como indivíduo de uma população, tenha uma boa ou má avaliação.

TIPOS DE REPRESENTAÇÃO

É possível representar as redes neurais de duas formas, uma é a representação direta onde o genótipo (representação da rede no cromossomo) contém todas as informações da rede como os pesos, as ligações, o número de neurônios e outras características, desta forma é possível recriar a rede sem processos de codificação e decodificação.

A outra forma é a representação indireta, onde codifica-se a rede para operá-la com o algoritmo genético, e decodifica-se para tê-la como rede novamente.

TIPOS DE EVOLUÇÃO

Uma rede neural melhora sua resposta ao treinamento por evolução quando se utiliza algoritmos genéticos, essa evolução pode se dar por diferença na estrutura da rede. A evolução por conexões ocorre quando se tem na população redes onde a única diferença entre elas é o número de ligações entre neurônios de uma camada e outra. Desse ponto de vista tenta-se evoluir através de combinações das conexões.

Outra forma de evolução é por quantidade de neurônios, e ocorre quando se tem redes na população onde a única diferença entre elas é o número de neurônios de cada camada oculta. Desse ponto de vista tenta-se evoluir através de combinações da quantidade de neurônios nas camadas ocultas.

A evolução por ajustes de pesos ocorre quando se tem na população, redes onde a única diferença entre elas são os pesos sinápticos. Nesta tenta-se evoluir através de cruzamentos e mutações dos indivíduos alterando assim seus pesos sinápticos.

Mas também pode-se tentar uma evolução com base em quaisquer características, da rede, sabendo como representar suas características, há a possibilidade de fazer evoluções levando em consideração as funções de ativação, as conexões cíclicas e paralelas, a quantidade de camadas, ou o que for interessante representar.

2.7 EXEMPLIFICAÇÃO DO PROCESSAMENTO DE REDE NEURAL

O único objetivo desta parte do trabalho é demonstrar como uma rede neural artificial processa as informações. Para isto considera-se que esta rede já está treinada, foi utilizada a resolução do XOR por se tratar de uma divisão espacial não-linearmente separável 2.3.

Suponha uma rede neural artificial para operar como a porta lógica XOR, também chamada porta OU-Exclusivo, que trabalha com binário, ou seja zeros e uns. Esta porta lógica trabalha com um par de numeros binários de 1 algarismo e seu resultado é um número binário de 1 algarismo. Esta porta é ativada quando somente uma das entradas é ativada, assim o resultado em 0 quando cada número do par tem o mesmo valor e 1 caso contrário, as combinações podem ser vista na Tabela 2.2.

O objetivo desta parte do trabalho é a exemplificação do processamento de uma rede e para isso será utilizada uma rede com 3 camadas entrada, oculta e saída, respectivamente contendo

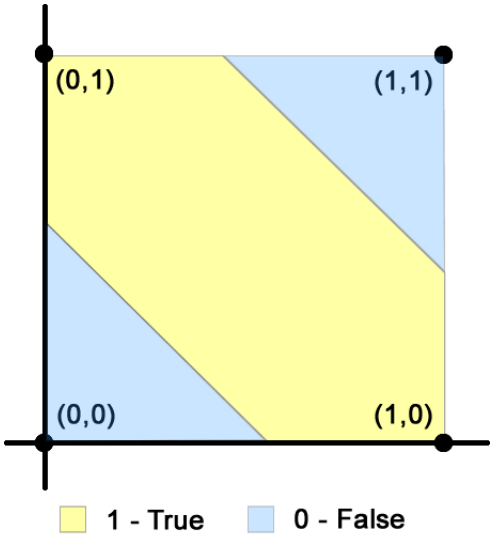


Figura 2.3: XOR: Gráfico

Binário 1	Binário 2	Binário 1 XOR Binário 2
0	0	0
1	0	1
0	1	1
1	1	0

Tabela 2.2: XOR: Combinações

2, 2 e 1 neurônios. Os neurônios utilizam função de ativação 'Passo' conforme a Figura 2.4, para melhor entendimento nomearemos os neurônios da seguinte forma: N (de neurônio) + número de ordem da camada + numero de ordem do neurônio na camada. Então, os neurônios da camada de entrada são $N11$ e $N12$, os da camada oculta são $N21$ e $N22$ e o da camada de saída é $N31$, como visto na Figura 2.5. Como foi dito, esta rede já está treinada e os valores das conexões sinápticas podem ser vistas no grafo da Figura 2.6.

$$f(s) = \begin{cases} +1 & \Rightarrow s \geqslant 0 \\ 0 & \Rightarrow s < 0 \end{cases}$$

Figura 2.4: XOR: Função de Ativação Passo

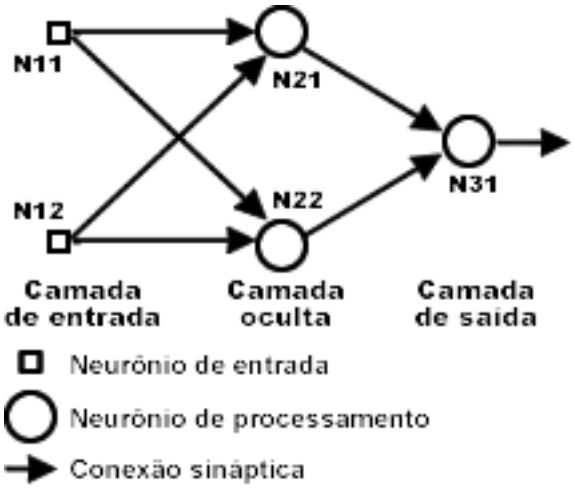


Figura 2.5: XOR: Representação do Neurônio Artificial

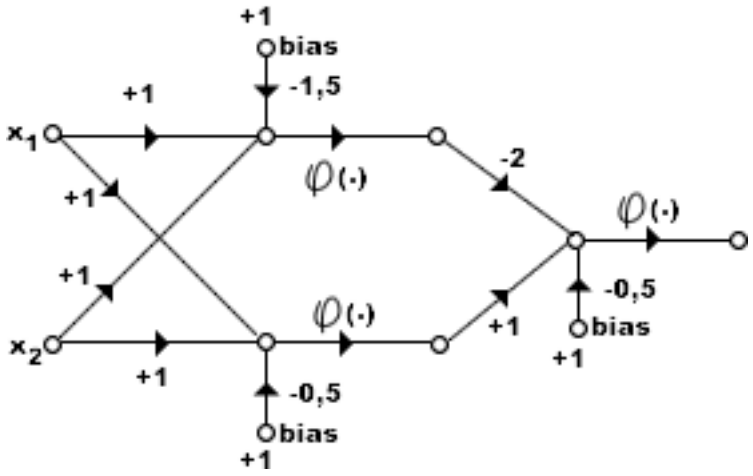


Figura 2.6: XOR: Grafo do Neurônio Artificial

Sendo assim podemos visualizar como a rede processa a informação e consegue nos responder de forma correta, será percebido que o processamento da rede nada mais é do que um conjunto de operações matemáticas, x_1 e x_2 são as entradas da rede e o valor de cada um pode ser 0 ou 1, conforme a regra do XOR mencionada anteriormente. Supondo a operação XOR para a combinação de entrada 1 e 0, então $x_1 = 1$ e $x_2 = 0$. A primeira operação será a multiplicação, seguida da soma e, por último a função de ativação. Como há três neurônios de processamento, e para facilitar as operações, elas serão feitas em três partes que podem ser vistas nos grafos das Figuras 2.7, 2.8 e 2.9.

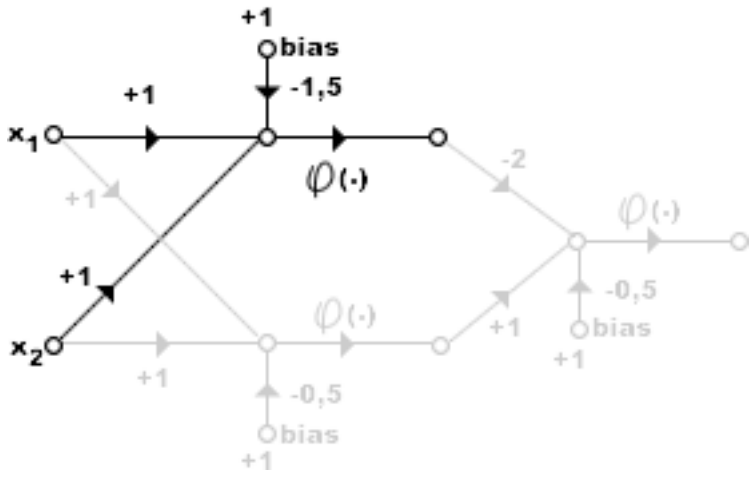


Figura 2.7: XOR - Operações de Neurônio N21: $((+1)*(+1))+((0)*(+1))+((+1)*(-1,5))=-0,5$; $-0,5$ na função de ativação Passo é 0

Na primeira parte é multiplicado a entrada de x_1 que no caso é 1 por $+1$ que é valor da conexão que liga $N11$ com $N21$ resultando $+1$, multiplica-se a entrada de x_2 que é 0 por $+1$ que é o valor da conexão que liga $N12$ com $N21$ resultando 0 e multiplica-se o valor do bias de $N21$ que é $+1$ por $-1,5$ que é o valor da conexão que liga o bias de $N21$ com $N21$ resultando $-1,5$, após isso é somado os três resultados totalizando $-0,5$, este total é passado pela função de ativação de $N21$ resultando 0, essas operações estão representadas na Figura 2.7.

Na segunda parte é multiplicado a entrada de x_1 que no caso é 1 por $+1$ que é valor da conexão que liga $N11$ com $N22$ resultando $+1$, multiplica-se a entrada de x_2 que é 0 por $+1$ que é o valor da conexão que liga $N12$ com $N22$ resultando 0 e multiplica-se o valor do bias de $N22$ que é $+1$ por $-0,5$ que é o valor da conexão que liga o bias de $N22$ com $N22$ resultando $-0,5$, após isso é somado os três resultados totalizando $+0,5$, este total é passado pela função de ativação de $N22$ resultando 1, essas operações estão representadas na Figura 2.8.

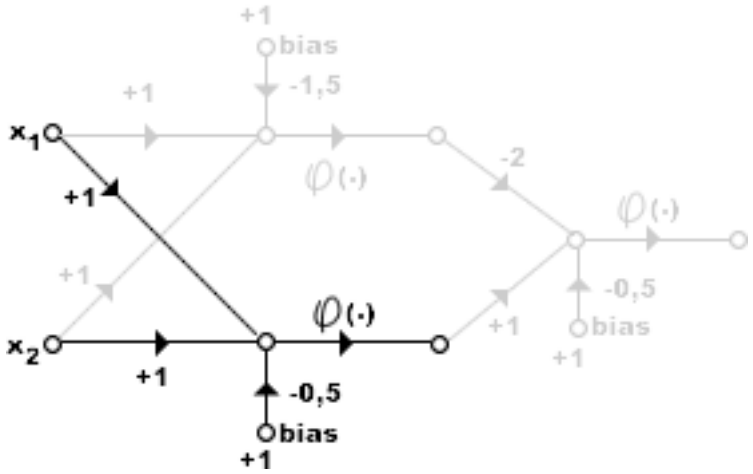


Figura 2.8: XOR - Operações de Neurônio N22: $((+1)*(+1))+((0)*(+1))+((+1)*(-0,5))=+0,5$; $+0,5$ na função de ativação Passo é $+1$

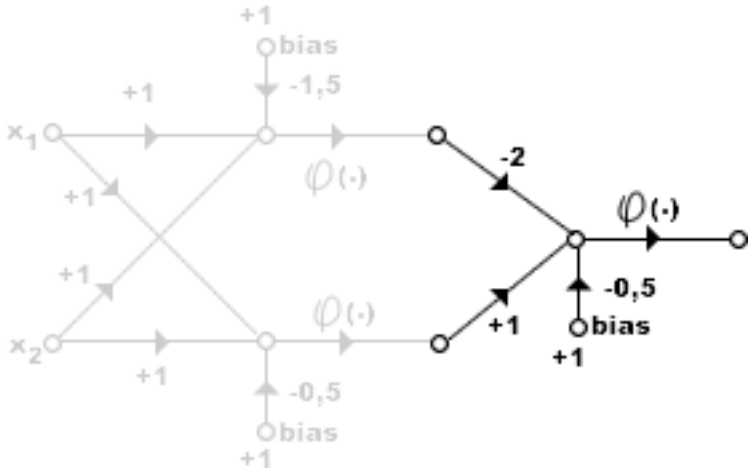


Figura 2.9: XOR - Operações de Neurônio N31: $((0)*(-2))+((+1)*(+1))+((+1)*(-0,5))=+0,5$; $+0,5$ na função Passo é $+1$

Agora, na terceira parte, as operações se repetem para N31, é multiplicado a saída de $N21$ que no caso é 0 por -2 que é valor da conexão que liga $N21$ com $N31$ resultando 0, multiplica-se a saída de $N22$ que é $+1$ por $+1$ que é o valor da conexão que liga $N22$ com $N31$ resultando $+1$ e multiplica-se o valor do bias de $N31$ que é $+1$ por $-0,5$ que é o valor da conexão que liga o bias de $N31$ com $N31$ resultando $-0,5$, após isso é somado os três resultados totalizando $+0,5$, este total é passado pela função de ativação de $N31$ resultando $+1$, essas operações estão representadas na Figura 2.9.

Esta rotina de processamento foi apresentada somente para o caso de $x_1 = 1$ e $x_2 = 0$, contudo funcionará para todos os outros, e funcionando para o objetivo, diz-se que a rede está treinada. A estrutura de uma rede faz com que ela seja capaz ou não de aprender, pois, a falta de neurônios ou camadas ocultas farão com que a rede não consiga aprender e o excesso fará

com que ela demore a aprender. Já o treinamento de uma rede faz com que os parâmetros livres, ou valores das conexões, sejam alterados para que a rede opere conforme um objetivo. No caso apresentado a rede precisava ter o comportamento da porta lógica XOR, e, são os algoritmos de aprendizagem os responsáveis por modificar os parâmetros livres.

3 APLICAÇÃO

A aplicação desenvolvida consiste em um jogo chamado Jogo da Velha 3D em Pinos, que é uma variação do famoso Jogo da Velha. O Jogo da Velha 3D em Pinos é muito parecido com o Jogo da Velha, pois é a combinação em linha reta de três símbolos do mesmo tipo que determina a pontuação. É descrito posteriormente o motivo da nomeclatura '3D em Pinos'.

3.1 O JOGO

O Jogo da Velha 3D em Pinos é um jogo de tabuleiro para ser jogado por duas pessoas, uma sendo o adversário da outra. Ele consiste em jogadas de pontuação em turnos, ou seja, cada um dos oponentes tem a sua vez de jogar e em sua vez ele tem a chance de marcar pontos. O tabuleiro é de madeira e tem o formato geométrico quadrado (ou similar) em sua superfície, essa superfície é dividida em 9 partes de mesma área, contabilizando assim 9 pequenos quadrados, conforme a Figura 3.1. No centro de cada quadrado há um pino cilíndrico, contabilizando assim 9 pinos distribuídos uniformemente na superfície. O jogo contém, além do tabuleiro, 24 peças de madeira com a forma de argola e tamanhos iguais, este formato é para que as peças possam ser encaixadas no pinos. As peças são divididas em duas cores, contabilizando assim, 12 peças para cada jogador.

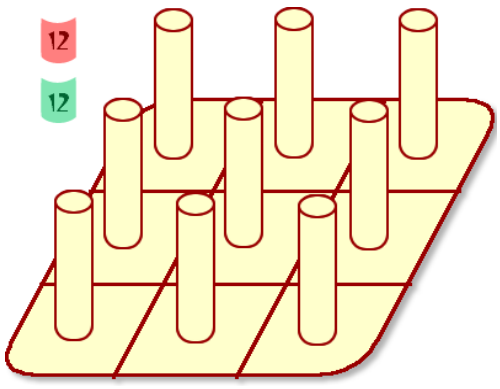


Figura 3.1: Jogo da velha 3d em pinos: Tabuleiro

Normalmente, em jogos de tabuleiros de 2 adversários, definimos que as peças claras são ditas como brancas e as escuras são ditas como pretas; outra definição é que o adversário das peças brancas sempre começa o jogo. Em cada pino do tabuleiro só podem ser encaixadas três peças, dando assim, um total de 27 posições possíveis para se encaixar as argolas. A colocação das argolas em cada pino seguem uma ordem de pilha, então a primeira argola colocada em um pino sempre será a mais próxima do tabuleiro, e a última argola colocada no mesmo pino será a mais afastada.

Como no Jogo da Velha, a pontuação é feita quando um jogador coloca 3 de suas peças em linha reta. No Jogo da Velha há 9 locais onde podemos colocar uma peça *X* (xis) ou *O* (bola) como na Figura 3.2, esses locais são chamados de casas e estas numeradas de 1 a 9 conforme a Figura 3.3.

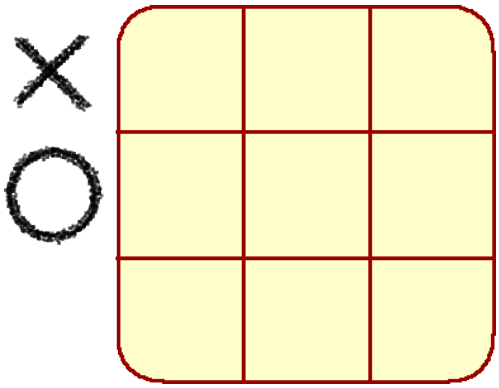


Figura 3.2: Jogo da velha: Tabuleiro

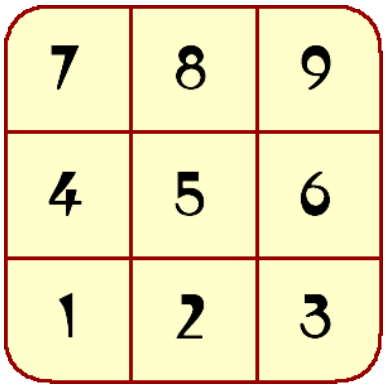


Figura 3.3: Jogo da velha: Casas

Como no Jogo da Velha temos 8 possibilidades de pontuação, podemos escrever todas as combinações que devem ser realizadas para a marcação de pontos como na Tabela 3.1 e visualizar a representação gráfica na Figura 3.4.

(1,2,3),(1,4,7),(1,5,9),(2,5,8), (3,5,7),(3,6,9),(4,5,6),(7,8,9)

Tabela 3.1: Combinações de pontuação do Jogo da Velha

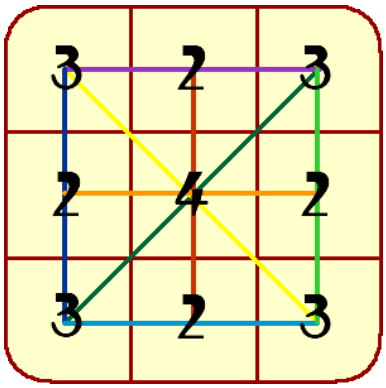


Figura 3.4: Jogo da velha: Combinações

No Jogo da Velha o primeiro a marcar um ponto ganha, como o exemplo da Figura 3.5 onde o jogador da peça O (bola) ganha o jogo colocando três peças em diagonal, após a vitória são retiradas as peças e o jogo começa novamente.

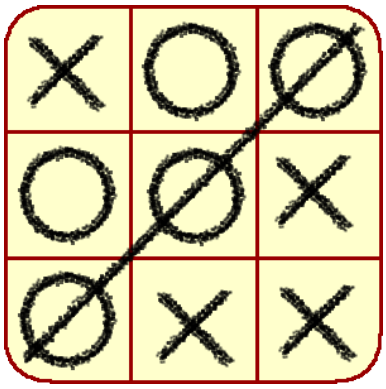


Figura 3.5: Jogo da velha: Exemplo de pontuação

Já no Jogo da Velha 3D em Pinos, os jogadores jogam até que se esgotem as 24 peças e no final são contabilizados os pontos marcados de acordo com as combinações feitas por cada jogador, veja o exemplo da Figura 3.6.

No Jogo da Velha 3D em Pinos há 49 possibilidades de pontuação, como temos 9 pinos e em cada pino podemos colocar 3 argolas, vamos considerar os pinos numerados de 1 a 9 como fizemos no Jogo da Velha tradicional, então a posição dos pinos ficaria como apresentado na Figura 3.7.

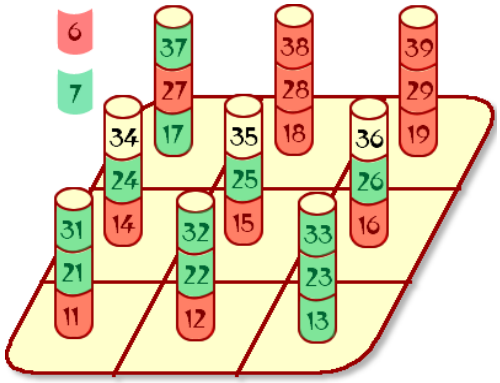


Figura 3.6: Jogo da velha 3d em pinos: Exemplo de pontuação

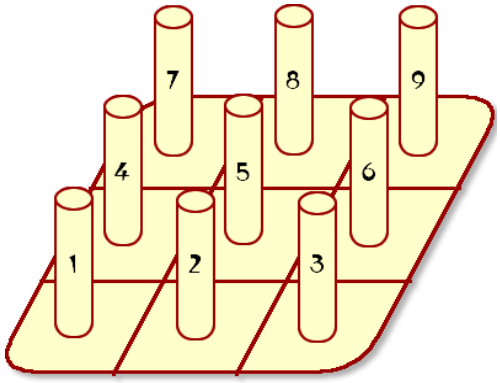


Figura 3.7: Jogo da velha 3d em pinos: Posicão dos pinos

Agora temos os pinos numerados, mas precisamos de saber a posição de cada argola no tabuleiro. Já sabemos que cada pino aceita 3 argolas, então vamos considerar que cada pino tem 3 níveis: 1, 2 e 3, sendo que o nível 1 é o mais próximo ao tabuleiro e o 3 o mais distante. Conseqüentemente o nível 1 deve ser preenchido para que se possa colocar argola no nível 2 e o nível 2 deve ser preenchido para que se possa colocar argola no nível 3. De acordo com isto, os níveis do pino são representados conforme a Figura 3.8.

Figura 3.8: Jogo da velha 3d em pinos: Níveis do pino

Com esse mapeamento que fizemos sobre os pinos e seus níveis, podemos mapear todas as posições possíveis do tabuleiro de uma forma bem simples e intuitiva. Cada posição do tabuleiro será composto de um número de dois algarismos, onde a a dezena (primeiro algarismo) representa os níveis do pino e a unidade (segundo algarismo) representa os pinos, sendo assim a argola no nível 1 do pino 1 é a 11 e a argola no nível 3 do pino 5 é a 35 como apresentado na Figura 3.9.

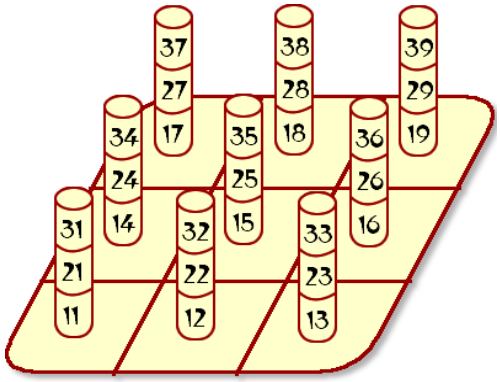


Figura 3.9: Jogo da velha 3d em pinos: Casas

Como já é possível ter a localização de qualquer argola no tabuleiro, pode-se escrever as 49 combinações de pontuação do Jogo da Velha 3D em Pinos conforme a Tabela 3.2 e visualizar graficamente um exemplo de combinação para a casa 33 na Figura 3.10.

(11,12,13),(11,14,17),(11,15,19),(11,21,31),(11,22,33),(11,24,37),(11,25,39), (21,22,23),(21,24,27),(21,25,29),(31,32,33),(31,34,37),(31,35,39),(31,22,13), (31,24,17),(31,25,19),(12,15,18),(12,22,32),(12,25,38),(22,25,28),(32,35,38), (32,25,18),(13,15,17),(13,16,19),(13,23,33),(13,25,37),(13,26,39),(23,25,27), (23,26,29),(33,35,37),(33,36,39),(33,25,17),(33,26,19),(14,15,16),(14,25,36), (14,24,34),(24,25,26),(34,35,36),(34,25,16),(15,25,35),(16,26,36),(17,18,19), (17,27,37),(17,28,39),(27,28,29),(37,38,39),(37,28,19),(18,28,38),(19,29,39)

Tabela 3.2: Combinações de pontuação do Jogo da Velha 3D em Pinos

3.2 LINGUAGEM DE PROGRAMAÇÃO

Por falta de conhecimento em outras linguagens de programação, foram escolhidas três como possibilidades para implementar a aplicação Jogo da Velha 3D em Pinos: C, Java e MatLab. C e Java por conhecimentos prévios da linguagem e MatLab por seu vasto recurso

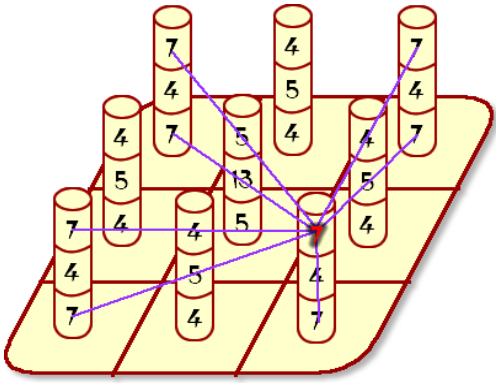


Figura 3.10: Jogo da velha 3d em pinos: Combinações para casa 33

matemático e por orientação no projeto de pesquisa devido às bibliotecas para trabalhar com redes neurais artificiais.

Em C houve dificuldade em localizar bibliotecas ou frameworks bem documentados para trabalhar com redes neurais, descartando-o. Em Java, após longa procura, foi localizado um interessante framework projetado e desenvolvido por João Ricardo Bittencourt (bolsista da iniciação científica CNPq/UNISINOS) com orientação do Prof. Dr. Fernando Osório, o ANNeF (Artificial Neural Networks Framework): é um framework de simples utilização e com ótima documentação. Tal framework foi muito estudado, e apresentava-se como boa ferramenta para implemetação, a implementação foi fácil e razoavelmente rápida, contudo seu treinamento inicial foi complicado e trabalhoso. Em MatLab, havia uma toolbox (no MatLab as bibliotecas são chamadas de Toolbox) chamada nnet (Neural Network Toolbox). Esta toolbox foi estudada e apresentou ótima documentação, com diversas funções de ativação implementadas, diversos modelos de redes, funções de simulação e de treinamento. Pelas facilidades, ela foi a opção mais viável para este estudo.

3.3 A REDE

A rede tem como objetivo aprender a jogar o Jogo da Velha 3D em Pinos, e para isso ela terá que aprender as regras do jogo, armar jogadas, pontuar e bloquear pontos. A estrutura da rede contém 3 camadas, uma camada de entrada que contêm 27 neurônios, uma camada oculta que contêm 49 neurônios e uma camada de saída que contêm 9 neurônios. Foi utilizada conexão plena entre a camada de entrada e a oculta e entre a camada oculta e a de saída, as ligações do tipo forward que têm o fluxo no sentido da camada anterior para a posterior e para função de ativação dos neurônios foi utilizada uma função hiperbólica. A dinâmica da rede é acíclica, isso quer dizer que não há ciclos no fluxo do processamento da rede e sua implementação foi no

software chamado MatLab.

O porquê da escolha dessa estrutura foi por um motivo simples: O jogador escolhe em qual dos 9 pinos quer colocar uma peça, então são 9 neurônios de saída. Sua decisão é sobre as peças já colocadas nos 9 pinos e, como não sabemos quantas peças foram colocadas em cada um dos nove pinos, entramos com todas as 27 possibilidades. São 27 pelo motivo de serem 9 pinos e somente no máximo 3 peças por pino, e isso resulta em um total de 27 ($3x9$). São 49 neurônios de camada oculta por serem 49 as possibilidades de combinação para pontuação. A conexão é plena pelo fato de que a escolha de um pino leva em consideração praticamente todas as 27 posições do tabuleiro. Foi escolhida uma função de ativação hiperbólica para restringir os valores entre -1 e 1, e o fluxo do tipo forward sem ciclos para simplificar o treinamento. No treinamento foi utilizado o algoritmo de aprendizagem por correção de erro também conhecido como Backpropagation, pois encaixa-se perfeitamente à estrutura da rede.

No Jogo da Velha 3D em Pinos, cada posição pode estar em um dos três estados: sem peça, com a minha peça ou com a peça do oponente. Para isso serão contabilizadas todas as posições, para cada posição há um neurônio responsável na camada de entrada e cada neurônio receberá um valor. Este valor pode ser, 0 quando não há peça para aquele neurônio, 1 quando há peça para aquele neurônio e a peça é minha e -1 quando há peça para aquele neurônio e a peça é do oponente. As posições onde as peças se encaixam vão de 1 a 27 em uma contagem seqüencial, mas como dito anteriormente optaremos pela numeração levando em consideração seu nível, sendo assim teremos peças do primeiro nível numeradas de 11 a 19, de segundo nível numeradas de 21 a 29 e de terceiro nível numeradas de 31 a 39, a Tabela 3.3 apresenta a numeração dos neurônios.

Como é preciso fazer com que a rede aprenda a jogar e, a intenção é que ela aprenda a jogar sem a intervenção de um humano para que o treinamento não seja demorado, criamos outras três redes para que pudessem ser oponentes: um clone, uma com 129 neurônios na camada oculta e outra com duas camadas ocultas com 129 e 9. O número 129 foi escolhido pois é a soma do número de influências (Tabelas 3.6 e 3.7) de cada posição. Todas as outras 3 redes seguem a características da rede principal, seus neurônios possuem função de ativação hiperbólica e as camadas têm conexão plena sem ciclos. O treinamento foi realizado com o Algoritmo de Backpropagation. As redes serão identificadas como R2D2, C3PO, Wall-e e Número 5 e seu comparativo arquitetural pode ser notado na Tabela 3.4.

Agora as redes têm oponentes para jogar durante horas e horas, mas mesmo assim ainda não sabem como jogar, nenhuma delas conhece a regra e, para que uma delas passe algum conhecimento às outras, pelo menos uma deveria conhecer o jogo. Foi criado então um juiz

Posição	Nível	Pino	Neurônio	Neurônio
1	1	1	11	111
2	1	2	12	112
3	1	3	13	113
4	1	4	14	121
5	1	5	15	122
6	1	6	16	123
7	1	7	17	131
8	1	8	18	132
9	1	9	19	133
10	2	1	21	211
11	2	2	22	212
12	2	3	23	213
13	2	4	24	221
14	2	5	25	222
15	2	6	26	223
16	2	7	27	231
17	2	8	28	232
18	2	9	29	233
19	3	1	31	311
20	3	2	32	312
21	3	3	33	313
22	3	4	34	321
23	3	5	35	322
24	3	6	36	323
25	3	7	37	331
26	3	8	38	332
27	3	9	39	333

Tabela 3.3: Posições no tabuleiro do Jogo da Velha 3D em Pinos

Característica	R2D2	C3PO	Wall-e	Número 5
Func. Ativação	Hiperbólica	Hiperbólica	Hiperbólica	Hiperbólica
Camadas	27x49x9	27x49x9	27x129x9	27x129x9x9
Tipo Conexão	Forward	Forward	Forward	Forward
Tipo Conexão	Forward	Forward	Forward	Forward
Grau Conectiv.	Plena	Plena	Plena	Plena
Dinâmica	Acíclica	Acíclica	Acíclica	Acíclica
Metod. Aprend.	Supervisionada	Supervisionada	Supervisionada	Supervisionada
Alg. Aprend.	Backpropagation	Backpropagation	Backpropagation	Backpropagation
Implementação	Software	Software	Software	Software

Tabela 3.4: Características das Redes de Treinamento

conhecedor das regras do jogo para auxiliar as redes enquanto elas disputam.

3.4 O JUIZ

Bom, mas como as redes estão aprendendo a jogar se apenas estão tentando jogar umas com as outras e nenhuma delas sabe as regras do jogo e nem quando fez ponto ou não? Nesse momento entra o juiz. Seu papel é informar qual jogada é melhor em relação a cada situação do jogo. A rede não sabe quando fez ponto ou bloqueou ponto do adversário, mas o juiz diz a ela a melhor jogada para fazer ou bloquear ponto. Assim, a rede vai aprendendo a marcar e bloquear pontos. A rede aprende a jogar pois está se ajustando, através do juiz, a fazer pontos, bloquear pontos do oponente e não jogar em lugares ruins.

A cada turno o juiz avalia a situação do jogo e a escolha da rede. Temos 9 pinos, cada um com 3 posições, dando um total de 27 posições, contudo a rede não escolhe em que posição vai jogar e sim em que pino vai jogar, então a rede escolhe um dos 9 pinos para jogar, e o juiz só precisa avaliar as 9 posições possíveis, para a melhor posição ele dá o valor de 1, para a pior ele dá o valor de -1, para as outras posições o valor é proporcional, menor que 1 e maior que -1. Mas como o juiz sabe a melhor posição em cada turno para a situação? O juiz varre as 9 posições. Para cada posição ele analisa as combinações possíveis e verifica se é possível fazer ponto (Ótimo), bloquear ponto do adversário (Ótimo), tentar caminho para um ponto na próxima jogada (Muito Bom), iniciar um caminho de ponto (Bom), colocar uma peça na combinação onde o adversário já tem uma peça (Ruim), colocar uma peça na combinação onde a rede e o adversário já tem uma peça finalizando assim as posições do pino e não fazendo pontuação alguma (Muito Ruim), e por último, colocar uma peça num pino sem espaço vago (Péssimo). O juiz também analisa a posição de onde se está fazendo a jogada, quantas combinações são possíveis fazer com aquela posição mesmo que no momento não façam pontos, isso porque é importante pegar posições estratégicas. O juiz pode analisar o jogo pontuar cada jogada pois ele tem conhecimento sobre todas as possibilidades de pontuação e as posições de influência. Atualmente o juiz considera apenas a situação atual do jogo para criticar a rede, não levando em consideração jogadas passadas e movimentações futuras, por isso é provável que a rede aprenda a cair em armadilhas. O Algoritmo do Juiz (Algoritmo 4) é o responsável por qualificar a jogada da rede e utiliza o Algoritmo 2 e o Algoritmo 3 como auxílio buscando neles as combinações possíveis e as porcentagens das posições de influência respectivamente.

3.4.1 ALGORITMO DO JUIZ

Saída: *comb*
inicio
| $comb = [(1,2,3),(1,4,7),(1,5,9),(1,10,19),(1,11,21),(1,13,25),(1,14,27),$
| $(10,11,12),(10,13,16),(10,14,18),(19,20,21),(19,22,25),(19,23,27),(19,11,3),$
| $(19,13,7),(19,14,9),(2,5,8),(2,11,20),(2,14,26),(11,14,17),(20,23,26),$
| $(20,14,8),(3,5,7),(3,6,9),(3,12,21),(3,14,25),(3,15,27),(12,14,16),$
| $(12,15,18),(21,23,25),(21,24,27),(21,14,7),(21,15,9),(4,5,6),(4,14,24),$
| $(4,13,22),(13,14,15),(22,23,24),(22,14,6),(5,14,23),(6,15,24),(7,8,9),$
| $(7,16,25),(7,17,27),(16,17,18),(25,26,27),(25,17,9),(8,17,26),(9,18,27)]$;
fim

Algoritmo 2: Combinações de Jogos

Saída: *perc*
inicio
| $perc = [12,6,12,6,8,6,12,6,12,$
| $7,9,7,9,25,9,7,9,7,$
| $14,8,14,8,10,8,14,8,14]$;
| $perc =$ cada elemento de $perc$ dividido por $max(perc)$;
fim

Algoritmo 3: Percentagens por Posição no Tabuleiro

```
Entrada: situacao;
Entrada: pino;
Entrada: jogador;
inicio
    pinospos = pinostira = pinosmarca = pinospontos = pinosperc = vetor[0,0,0,0,0,0,0,0,0,0];
    lcomb = Combinacoes(); lperc = Percentagens();
    para (k = 1 ; k <= 9 ; k++ ) faça
        se (pino k tem 3 peças ) então
            pinosperc[k] = −1;
        senão
            lposicao = primeira posição vaga do pino k;
            lgrau = ltira = lmarca = lpontos = 0;
            ltotalcomb = número total de combinações;
            para (i = 1 ; i <= ltotalcomb ; i++ ) faça
                cb1 = lcomb[i,1]; cb2 = lcomb[i,2]; cb3 = lcomb[i,3];
                ltem =((cb1 == lposicao)||(cb2 == lposicao)||(cb3 == lposicao));
                se (ltem ) então
                    pri = situacao[cb1]; seg = situacao[cb2]; ter = situacao[cb3];
                    lindice = ((abs(pri)+abs(seg)+abs(ter))∗2)+((pri+seg+ter)∗ jogador);
                    se lindice == 6 então
                        lgrau = lgrau+lperc(lposicao)∗4; lpontos = lpontos+1;
                        lmarca = lmarca+1;
                    senão se lindice == 2 então
                        lgrau = lgrau+lperc(lposicao)∗3.9; lpontos = lpontos+1;
                        ltira = ltira+1;
                    senão se lindice == 3 então
                        lgrau = lgrau+lperc(lposicao)/4;
                    senão se lindice == 0 então
                        lgrau = lgrau+lperc(lposicao)/16;
                    senão se lindice == 1 então
                        lgrau = lgrau−lperc(lposicao)/16;
                    senão se lindice == 4 então
                        lgrau = lgrau−lperc(lposicao)/8;
                    fim
                fim
            fim
            pinosperc[k] = lgrau; pinospontos[k] = lpontos; pinosmarca[k] = lmarca;
            pinostira[k] = ltira; pinospos[k] = lposicao;
        fim
    fim
    para (j = 1 ; j <= 9 ; j++ ) faça
        se pinosperc[j] == −1 então
            pinosperc(j) = −max(max(pinosperc),1);
        fim
    fim
    pinosperc = pinosperc/max(max(pinosperc),1);
    Resultado: [pinosperc, pinospontos, pinosmarca, pinostira, pinospos]
fim
```

Algoritmo 4: Algoritmo do Juiz

3.5 POSSIBILIDADES DE PONTUAÇÃO

Só existem 49 possibilidades de pontuação, então a rede deve aprender como jogar nas combinações conforme Tabela 3.5.

(11,12,13),(11,14,17),(11,15,19),(11,21,31),(11,22,33),(11,24,37),(11,25,39), (21,22,23),(21,24,27),(21,25,29),(31,32,33),(31,34,37),(31,35,39),(31,22,13), (31,24,17),(31,25,19),(12,15,18),(12,22,32),(12,25,38),(22,25,28),(32,35,38), (32,25,18),(13,15,17),(13,16,19),(13,23,33),(13,25,37),(13,26,39),(23,25,27), (23,26,29),(33,35,37),(33,36,39),(33,25,17),(33,26,19),(14,15,16),(14,25,36), (14,24,34),(24,25,26),(34,35,36),(34,25,16),(15,25,35),(16,26,36),(17,18,19), (17,27,37),(17,28,39),(27,28,29),(37,38,39),(37,28,19),(18,28,38),(19,29,39)

Tabela 3.5: Combinações no tabuleiro do Jogo da Velha 3D em Pinos

3.6 POSIÇÕES DE INFLUÊNCIA

As posições de influência são necessárias para que o juiz possa qualificar cada jogada que as redes fizerem. Quando uma rede escolhe o pino 1 para jogar pode ser que o pino esteja vazio (A peça entra na posição 11), pode ser que já tenha uma peça (A peça entra na posição 21), pode só ter uma vaga (A peça entra na posição 31) e pode ser que não tenha vaga (A peça não pode ser colocada neste pino). O juiz precisa saber quais são as combinações que influenciam ao jogar em cada uma das posições, ex: apesar de a posição 11 fazer combinação com a 21 e a 31, se podemos jogar a peça na posição 11 quer dizer que as posições 21 e 31 ainda não estão ocupadas. Então para a posição 11, as posições 21 e 31 não são válidas para critério de pontuação. E para a posição 21, a posição 11 é válida mas a 31 não é, formando uma combinação parcial de pontos. Agora para a posição 31, as posições 11 e 21 são válidas. Na Tabelas 3.6 e 3.7 seguem, em vermelho as combinações que não influenciam a posição e em verde as que influenciam parcialmente.

Posição	Relação de influência	Influências	Max Pontos	Combinações
11	12, 13, 14, 15, 17, 19, 22, 24, 25, 33, 37, 39	12	7	(12,13), (14,17), (15,19), (22,33), (24,37), (25,39), (21,31)
12	11, 13, 15, 18, 25, 38	6	4	(11,13), (15,18), (25,38), (22,32)
13	11, 12, 15, 16, 17, 19, 22, 25, 26, 31, 37, 39	12	7	(11,12), (15,17), (16,19), (22,31), (25,37), (26,39), (23,33)
14	11, 15, 16, 17, 25, 36	6	4	(11,17), (15,16), (25,36), (24,34)
15	11, 12, 13, 14, 16, 17, 18, 19	8	5	(11,19), (12,18), (13,17), (14,16), (25,35)
16	13, 14, 15, 19, 25, 34	6	4	(13,19), (14,15), (25,34), (26,36)
17	11, 13, 14, 15, 18, 19, 24, 25, 28, 31, 33, 39	12	7	(11,14), (13,15), (18,19), (24,31), (25,33), (28,39), (27,37)
18	12, 15, 17, 19, 25, 32	6	4	(12,15), (17,19), (25,32), (28,38)
19	11, 13, 15, 16, 17, 18, 25, 26, 28, 31, 33, 37	12	7	(11,15), (13,16), (17,18), (25,31), (26,33), (28,37), (29,39)

Tabela 3.6: Posições que podem influenciar na jogada. Parte I

Posição	Relação de influência	Influências	Max Pontos	Combinações
21	11, 22, 23, 24, 25, 27, 29	7	4	(22,23), (24,27), (25,29), (11,31)
22	11, 12, 13, 21, 23, 25, 27, 31, 33	9	5	(21,23), (25,28), (11,33), (13,31), (12,32)
23	13, 21, 22, 25, 26, 27, 29	7	4	(21,22), (25,27), (26,29), (13,33)
24	11, 14, 17, 21, 25, 26, 27, 31, 37	9	5	(21,27), (25,26), (11,37), (17,31), (14,34)
25	11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 26, 27, 28, 29, 31, 32, 33, 34, 36, 37, 38, 39	25	13	(21,29), (22,28), (23,27), (24,26), (11,39), (12,38), (13,37), (14,36), (16,34), (17,33), (18,32), (19,31), (15,35)
26	13, 16, 19, 23, 24, 25, 29, 33, 39	9	5	(23,29), (24,25), (13,39), (19,33), (16,36)
27	17, 21, 23, 24, 25, 28, 29	7	4	(21,24), (23,25), (28,29), (17,37)
28	17, 18, 19, 22, 25, 27, 29, 37, 39	9	5	(22,25), (27,29), (17,39), (19,37), (18,38)
29	19, 21, 23, 25, 26, 27, 28	7	4	(21,25), (23,26), (27,28), (19,39)
31	11, 13, 17, 19, 21, 22, 24, 25, 32, 33, 34, 35, 37, 39	14	7	(32,33), (34,37), (35,39), (22,13), (24,17), (25,19), (11,21)
32	12, 18, 22, 25, 31, 33, 35, 38	8	4	(31,33), (35,38), (25,18), (12,22)
33	11, 13, 17, 19, 22, 23, 25, 26, 31,32,35,36,37,39	14	7	(31,32), (35,37), (36,39), (22,11), (25,17), (26,19), (13,23)
34	14, 16, 24, 25, 31, 35, 36, 37	8	4	(31,37), (35,36), (25,16), (14,24)
35	15, 25, 31, 32, 33, 34, 36, 37, 38,39	10	5	(31,39), (32,38), (33,37), (34,36), (15,25)
36	14, 16, 25, 26, 33, 34, 35, 39	8	4	(33,39), (34,35), (25,14), (16,26)
37	11, 13, 17, 19, 24, 25, 27, 28, 31,33,34,35,38,39	14	7	(31,34), (33,35), (38,39), (24,11), (25,13), (28,19), (17,27)
38	12, 18, 25, 28, 32, 35, 37, 39	8	4	(32,35), (37,39), (25,12), (18,28)
39	11, 13, 17, 19, 25, 26, 28, 29, 31,33,35,36,37,38	14	7	(31,35), (33,36), (37,38), (25,11), (26,13), (28,17), (19,29)

Tabela 3.7: Posições que podem influenciar na jogada. Parte II

Agora graficamente na Figura 3.11, note que para escolher a casa 26 (a casa com o número nove em vermelho) para jogar é necessário observar as casas 13, 23 e 33 no pino 3, 24 no pino 4, 25 no pino 5, 16 no pino 6 e 19, 29 e 39 no pino 9, pois todas elas formam combinações com a 26, repare também que apesar da casa 36 fazer combinação com 26, ela não deve ser levada em consideração, pois se a casa 36 estiver preenchida, a casa 26 também estará.

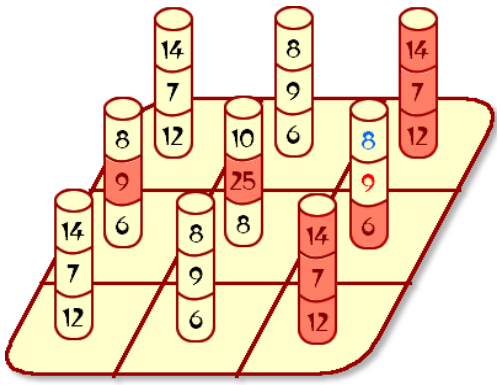


Figura 3.11: Jogo da velha 3d em pinos: Casas de influências para casa 26

3.7 SITUACAO DO TABULEIRO

A situação do tabuleiro informa onde está cada peça dos jogadores, as casas sem peças são representadas com o valor 0, as casas com peças do primeiro jogador são representadas com o valor 1 e as casas do segundo jogador são representadas com o valor -1, veja um exemplo na Figura 3.12. Pode-se dizer que o valor 1 representa o primeiro jogador 1 e o valor -1 representa o segundo jogador.

Para as redes a situação do tabuleiro tem apenas uma visão, nas casas com valor 0 não há peças, nas casas com valor 1 estão as peças dela e nas casas com valor -1 estão as peças do oponente. Então, para a primeira rede sua visão é a mesma da situação do tabuleiro, mas para a segunda é necessário inverter a situação do tabuleiro. Essa inversão pode ser conseguida multiplicando o valor que representa o jogador pela situação do tabuleiro, ocorrendo assim, apenas a inversão da situação para a segunda rede. Supondo a situação do tabuleiro conforme a Figura 3.12, a visão da situação do tabuleiro para cada uma das duas redes é representada pela Figura 3.13.

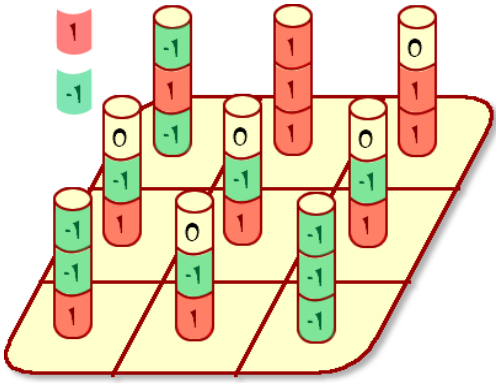
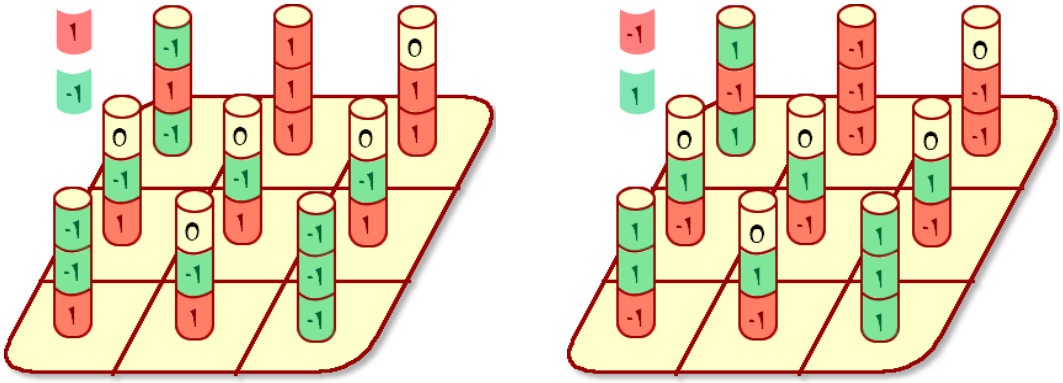


Figura 3.12: Jogo da velha 3d em pinos: Situação



(a) Primeiro jogador (b) Segundo jogador

Figura 3.13: Jogo da velha 3d em pinos: Visão da situação do tabuleiro

3.8 APLICAÇÃO EM PROGRAMAS

Como já foi dito anteriormente, foi utilizado MatLab pela agilidade na construção de redes neurais artificiais. O desenvolvimento da aplicação foi feito em duas partes, o objetivo da primeira parte foi a criação das quatro redes e seu treinamento, o arquivo principal desta parte é o jv3d_treinamento.m. Na segunda parte o objetivo foi disponibilizar uma forma de usuários humanos confrontarem as redes, o arquivo principal desta parte é o jv3d_jogo.m.

Na Figura 3.14 é apresentada a organização dos arquivos utilizados para o treinamento das redes, o arquivo jv3d_treinamento.m gerencia o treinamento: verifica se as quatro redes existem (jv3d_rna1_ff_27_49_9.mat, jv3d_rna2_ff_27_49_9.mat, jv3d_rna3_ff_27_129_9.mat e jv3d_rna4_ff_27_129_9_9.mat, respectivamente, R2D2, C3PO, Wall-e e Numero 5); se elas não existirem é utilizado os arquivos jv3d_ff_27_49_9.m para criar R2D2 e C3PO, jv3d_ff_27_129_9.m para criar Wall-e e jv3d_ff_27_129_9_9.m para criar Numero 5.

Para criar a estrutura de pontuação de cada rede é utilizado o arquivo jv3d_cria_jogador.m. O arquivo jv3d_rna_joga.m efetua as jogadas de cada rede e corrige seus pesos sinápticos

utilizando o arquivo jv3d_classificacao.m (juiz) para informar a jogada correta. O arquivo jv3d_situacao.m é utilizado para mudar a visão do tabuleiro como foi dito anteriormente.

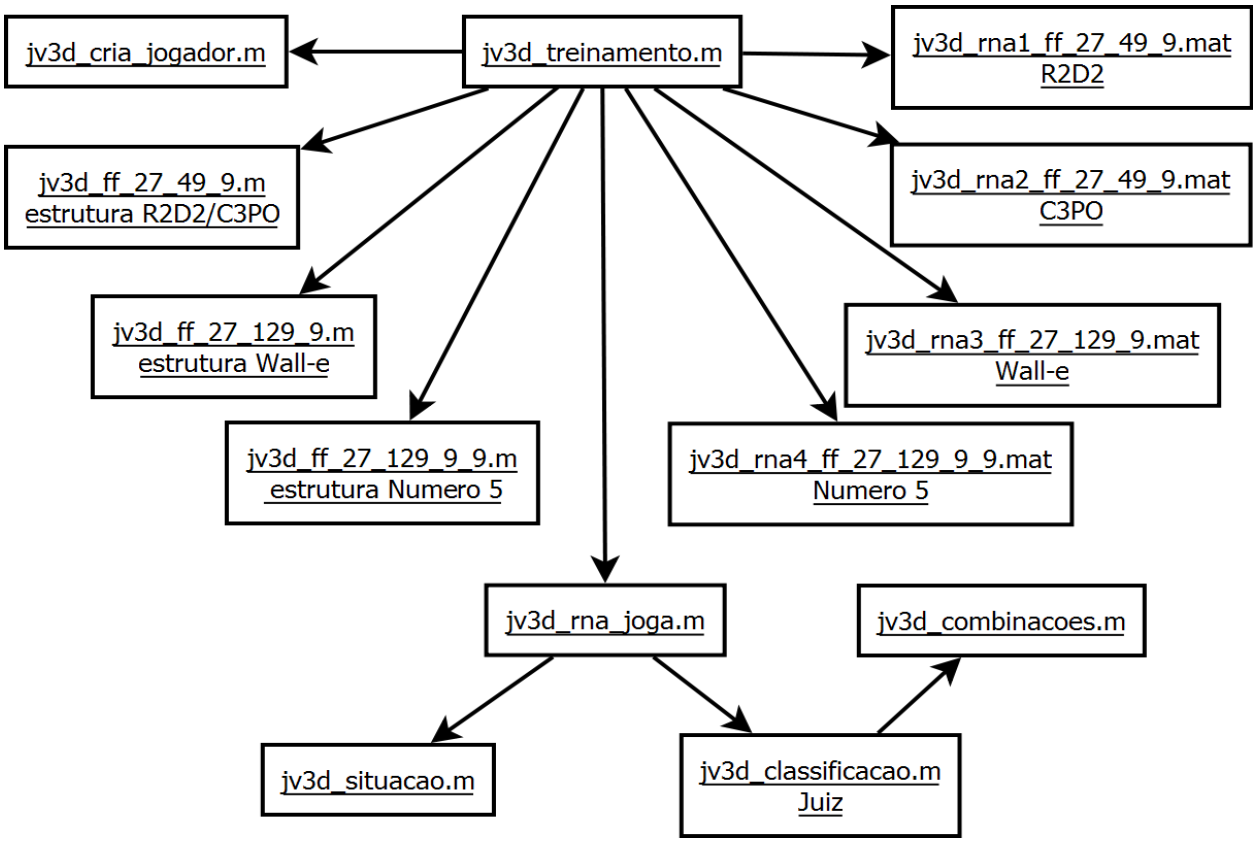


Figura 3.14: Jogo da velha 3d em pinos: Arquivos de treinamento

Na Figura 3.15, o arquivo jv3d_jogo.m gerencia o jogo, este jogo é gráfico e ao pressionar *i* é apresentado um menu com informações necessárias para começar a jogar. Neste programa somente duas redes estão disponíveis para serem confrontadas, R2D2 (jv3d_nra1_ff_27_49_9.mat) e C3PO (jv3d_nra2_ff_27_49_9.mat), além de ser possível ser jogado por duas pessoas. O arquivo jv3d_config.m é utilizado para armazenar as configurações do jogo, o arquivo jv3d_posicoes.m armazenas as coordenadas 2D de cada casa do tabuleiro. O arquivo jv3d_rna_joga.m efetua as jogadas e treina a rede em conjunto com o arquivo jv3d_classificacao.m (juiz). O arquivo jv3d_situacao.m é utilizado para mudar a visão do tabuleiro como foi dito anteriormente.

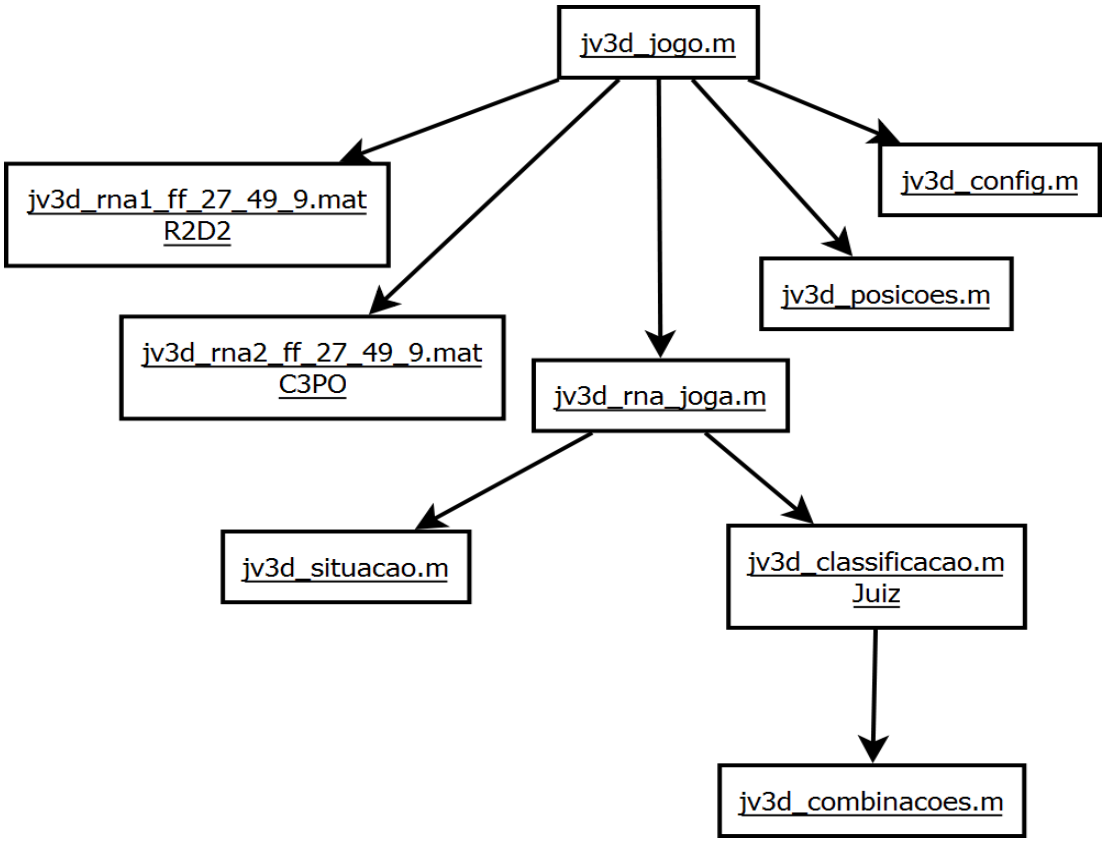


Figura 3.15: Jogo da velha 3d em pinos: Arquivos de jogo

3.9 TREINAMENTO

O treinamento foi iniciado em 03 de novembro de 2008 às 00 horas e 30 minutos com o "Treinamento I"descrito adiante. Nenhuma rede tem conhecimento prévio das regras do jogo e os treinamentos são cumulativos, as redes mantêm suas experiências anteriores a cada novo treino. Cada partida é composta de 24 turnos. Cada turno é a vez do jogador interagir com o jogo, nele é feita uma jogada. Os turnos são intercalados entre os jogadores. As partidas foram disputadas entre as quatro redes e ocorreram da seguinte forma C3PO x Wall-e, Wall-e x Número 5, Número 5 x R2D2, R2D2 x C3PO. Dessa forma, todas as redes tiveram a vez de jogar como o jogador das peças brancas e jogador das peças pretas. Contudo, as redes não jogaram com todos os outros participantes: dos três adversários de cada rede, ela jogou apenas com dois, sendo que contra um ela jogava com as peças brancas e com outro ela jogava com as peças pretas. O resultado dos jogos podem ser vitória, empate e derrota. A cada turno a rede tinha 25 chances de escolher um pino para jogar, se ela não conseguisse jogar em um pino vago era considerado que ela não-entendeu e obteve uma derrota e sua adversária ganhava o jogo. As quatro redes têm características muito semelhantes. Para a leitura dos treinamentos temos que levar em consideração que não houve os confrontos das redes R2D2 x Wall-e e C3PO x Número 5.

3.9.1 TREINAMENTO I

Ocorreram 2700 partidas em 13 horas, uma média de 3,5 partidas por minuto, a ordem das partidas foi C3PO x Wall-e, Wall-e x Número 5, Número 5 x R2D2, R2D2 x C3PO. As redes tiveram 25 chances de escolher um pino vago, mas mesmo assim podemos perceber que elas não-entenderam como jogar em muitas partidas, significando que elas ainda não aprenderam jogar. Wall-e é a que menos consegue jogar, é a que mais perdeu pelo status não-entendeu (quase 59% das partidas), é a que menos venceu e a que mais perdeu. Ainda podemos ver que apesar de R2D2 e C3PO serem clones antes do treinamento, suas experiências fizeram com que cada uma aprendesse a jogar de uma forma diferente. R2D2 teve confrontos com C3PO e Número 5, enquanto C3PO teve confrontos com R2D2 e Wall-e. Nos confrontos R2D2 x C3PO, R2D2 levou uma pequena vantagem em vitórias de 50.81% contra 46,07% de C3PO, praticamente empate técnico. Podemos ver que a rede que mais aprendeu foi a rede Número 5, teve o menor número de Não-entendimento e a maior marcação de pontos, o resultado deste treinamento pode ser visto nas Tabelas 3.8 e 3.9.

	Pontos	Não-entendeu	Vitórias	Empates	Derrotas
R2D2	5339	602	532	40	778
C3PO	5684	368	849	41	460
Wall-e	4316	796	277	33	1040
Número 5	6450	258	969	32	349

Tabela 3.8: Treinamento I: Pontuação Por Rede

	R2D2				C3PO				Wall-e				Número 5			
	V	E	D	P	V	E	D	P	V	E	D	P	V	E	D	P
R2D2	0	0	0	0	343	21	311	235	0	0	0	0	189	19	467	367
C3PO	311	21	343	263	0	0	0	0	538	20	117	105	0	0	0	0
Wall-e	0	0	0	0	117	20	538	433	0	0	0	0	160	13	502	363
Número 5	467	19	189	138	0	0	0	0	502	13	160	120	0	0	0	0

V → vitória; E → empate; D → Derrota; P → Não-entendeu

Tabela 3.9: Treinamento I: Pontuação Contra Redes

3.9.2 TREINAMENTO II

Agora ocorreram mais 200 partidas em 1 hora, uma média de 3,3 partidas por minuto. Foi invertida a ordem das redes jogarem uma com as outras: Wall-e x C3PO, C3PO x R2D2, R2D2 x Número 5, Número 5 x Wall-e. A inversão foi feita para observarmos se quem inicia o confronto leva vantagem ou vice-versa. Continuamos com 25 chances da rede escolher um pino vago. Podemos ver que C3PO não deu chances à R2D2: das 35 vitórias de C3PO sobre R2D2, 31 foi por que R2D2 Não-entendeu; das 11 vitórias de R2D2 sobre C3PO, 6 foi por que C3PO Não-entendeu. Podemos concluir que quando o jogo foi até o final, ou seja nenhuma delas Não-entendeu, R2D2 teve uma vitória a mais. Número 5 continuou com poucos Não-entendimentos, e agora foi a vez de C3PO ser a que mais aprendeu a jogar, com menos Não-entendimento, mas quem marcou mais pontos continuou sendo a Número 5. O resultado deste treinamento pode ser visto nas Tabelas 3.10 e 3.11.

	Pontos	Não-entendeu	Vitórias	Empates	Derrotas
R2D2	392	51	33	6	61
C3PO	416	21	65	7	28
Wall-e	373	50	34	5	61
Número 5	420	24	57	4	39

Tabela 3.10: Treinamento II: Pontuação Por Rede

	R2D2				C3PO				Wall-e				Número 5			
	V	E	D	P	V	E	D	P	V	E	D	P	V	E	D	P
R2D2	0	0	0	0	11	4	35	31	0	0	0	0	22	2	26	20
C3PO	35	4	11	6	0	0	0	0	30	3	17	15	0	0	0	0
Wall-e	0	0	0	0	17	3	30	23	0	0	0	0	17	2	31	27
Número 5	26	2	22	12	0	0	0	0	31	2	17	12	0	0	0	0

V → vitória; E → empate; D → Derrota; P → Não-entendeu

Tabela 3.11: Treinamento II: Pontuação Contra Redes

3.9.3 TREINAMENTO III

Agora ocorreram mais 100 partidas em 28 minutos, uma média de 3,6 partidas por minuto. Foi retornada a ordem normal das redes jogarem umas com as outras: R2D2 x C3PO, C3PO x Wall-e, Wall-e x Número 5, Número 5 x R2D2. Aumentamos em quatro vezes o grau de satisfação por fazer pontos, por bloquear pontos do adversário e aumentamos também em quatro vezes o grau de insatisfação por tentar jogar em um pino cheio. Continuamos com 25 chances da rede jogar em um pino vago. Com essas alterações podemos notar que as redes passaram a perder menos por Não-entendimento, com exceção da Wall-e que parece ser um pouco mais lenta em seu aprendizado: conforme as regras mudam ela se adapta de forma mais lenta que as outras, talvez pela quantidade de conexões que ela tem da camada oculta para a camada de saída. Com a ordem natural novamente R2D2 voltou a ganhar de C3PO, mas isso é empate técnico, diferença de uma vitória em 25 jogos. Número 5 continua a marcar mais pontos, o resultado deste treinamento pode ser visto nas Tabelas 3.12 e 3.13.

	Pontos	Não-entendeu	Vitórias	Empates	Derrotas
R2D2	196	2	12	7	31
C3PO	195	0	33	7	10
Wall-e	84	30	1	1	48
Número 5	198	0	46	1	3

Tabela 3.12: Treinamento III: Pontuação Por Rede

	R2D2				C3PO				Wall-e				Número 5			
	V	E	D	P	V	E	D	P	V	E	D	P	V	E	D	P
R2D2	0	0	0	0	10	6	9	1	0	0	0	0	2	1	22	1
C3PO	9	6	10	0	0	0	0	0	24	1	0	0	0	0	0	0
Wall-e	0	0	0	0	0	1	24	15	0	0	0	0	1	0	24	15
Número 5	22	1	2	0	0	0	0	0	24	0	1	0	0	0	0	0

V → vitória; E → empate; D → Derrota; P → Não-entendeu

Tabela 3.13: Treinamento III: Pontuação Contra Redes

3.9.4 TREINAMENTO IV

Agora ocorreram mais 100 partidas em 16 minutos, uma média de 6,2 partidas por minuto. Foi mantida a ordem normal das redes jogarem umas com as outras: R2D2 x C3PO, C3PO x Wall-e, Wall-e x Número 5, Número 5 x R2D2. Mantivemos a satisfação e insatisfação, como no resultado anterior, em quatro vezes para pontos e bloqueios de pontos e escolha de pino cheio. Fizemos uma nova alteração nas regras para que a rede apenas tenha uma chance de escolher um pino vago senão é contabilizado Não-entendimento. Podemos ver que as redes realmente aprenderam a jogar, com exceção de Wall-e que continua com um alto nível de Não-entendimento. Repare que por só terem uma chance de escolher um pino vago o número de partidas por minuto aumentou bem. A número 5 foi a única que não perdeu por Não-entendimento. C3PO e R2D2 perderam por Não-entendimento uma para a outra e continuam em empate técnico, mas agora com um pouco mais de vitória para C3PO, o resultado deste treinamento pode ser visto nas Tabelas 3.14 e 3.15.

	Pontos	Não-entendeu	Vitórias	Empates	Derrotas
R2D2	174	1	14	6	30
C3PO	137	1	37	2	11
Wall-e	102	34	7	3	40
Número 5	193	0	33	7	10

Tabela 3.14: Treinamento IV: Pontuação Por Rede

	R2D2				C3PO				Wall-e				Número 5			
	V	E	D	P	V	E	D	P	V	E	D	P	V	E	D	P
R2D2	0	0	0	0	10	2	13	1	0	0	0	0	4	4	17	0
C3PO	13	2	10	1	0	0	0	0	24	0	1	0	0	0	0	0
Wall-e	0	0	0	0	1	0	24	21	0	0	0	0	6	3	16	13
Número 5	17	4	4	0	0	0	0	0	16	3	6	0	0	0	0	0

V → vitória; E → empate; D → Derrota; P → Não-entendeu

Tabela 3.15: Treinamento IV: Pontuação Contra Redes

3.9.5 TREINAMENTO V

Agora ocorreram mais 6000 partidas em 18 horas, uma média de 5.3 partidas por minuto. Foi mantida a ordem normal das redes jogarem umas com as outras: R2D2 x C3PO, C3PO x Wall-e, Wall-e x Número 5, Número 5 x R2D2. Mantivemos a satisfação e insatisfação como no resultado anterior em quatro vezes para pontos e bloqueios de pontos e escolha de pino cheio, voltamos a regra de chance de escolha de pino vago para 25 chances. O objetivo deste treino foi saber se Wall-e reage e aprende as regras do jogo. Podemos perceber que número de vitórias entre elas se equilibrou bem e apesar de C3PO ser a que mais venceu, ela ficou em último em marcação de pontos. Número 5 é que marcou mais pontos. Wall-e parece ter aprendido bastante já que perdeu por Não-entendimento em somente 3,5% das partidas que fez. Podemos perceber que entre os confrontos quem ganhou mais vezes foi quem começou o jogo, o resultado deste treinamento pode ser visto nas Tabelas 3.16 e 3.17.

	Pontos	Não-entendeu	Vitórias	Empates	Derrotas
R2D2	11315	0	1221	477	1302
C3PO	10828	0	1398	465	1137
Wall-e	10937	106	1144	480	1376
Número 5	12227	0	1280	492	1228

Tabela 3.16: Treinamento V: Pontuação Por Rede

	R2D2				C3PO				Wall-e				Número 5			
	V	E	D	P	V	E	D	P	V	E	D	P	V	E	D	P
R2D2	0	0	0	0	720	249	531	0	0	0	0	0	501	228	771	0
C3PO	531	249	720	0	0	0	0	0	867	216	417	0	0	0	0	0
Wall-e	0	0	0	0	417	216	867	50	0	0	0	0	727	264	509	56
Número 5	771	228	501	0	0	0	0	0	509	264	727	0	0	0	0	0

V → vitória; E → empate; D → Derrota; P → Não-entendeu

Tabela 3.17: Treinamento V: Pontuação Contra Redes

3.9.6 TREINAMENTO VI

Agora ocorreram mais 200 partidas em 51 minutos, uma média de 3.9 partidas por minuto. Foi mantida a ordem normal das redes jogarem umas com as outras: R2D2 x C3PO, C3PO x Wall-e, Wall-e x Número 5, Número 5 x R2D2. Mantivemos a satisfação e insatisfação como no resultado anterior em quatro vezes para pontos e bloqueios de pontos e escolha de pino cheio, voltamos para a regra em que a rede apenas tem uma chance de escolher um pino vago senão é contabilizado Não-entendimento. Podemos ver que Wall-e aprendeu a jogar, pois não perdeu por Não-entendimento (fazendo com que aprendesse a ganhar partidas e pontuar), fez mais pontos e foi a segunda que mais venceu, mas foi a segunda que mais perdeu. Número 5 que era uma das que mais vencia e a que sempre fez mais pontos ficou em último em vitórias e em terceiro em pontos. Os jogadores que começaram as partidas ganharam mais vezes, com exceção dos confrontos Número 5 x R2D2, onde R2D2 ganhou mais vezes. C3PO fez menos pontos, o resultado deste treinamento pode ser visto nas Tabelas 3.18 e 3.19.

	Pontos	Não-entendeu	Vitórias	Empates	Derrotas
R2D2	387	0	48	19	33
C3PO	339	0	41	23	36
Wall-e	401	0	43	20	37
Número 5	371	0	29	16	55

Tabela 3.18: Treinamento VI: Pontuação Por Rede

	R2D2				C3PO				Wall-e				Número 5			
	V	E	D	P	V	E	D	P	V	E	D	P	V	E	D	P
R2D2	0	0	0	0	25	10	15	0	0	0	0	0	23	9	18	0
C3PO	15	10	25	0	0	0	0	0	26	13	11	0	0	0	0	0
Wall-e	0	0	0	0	11	13	26	0	0	0	0	0	32	7	11	0
Número 5	18	9	23	0	0	0	0	0	11	7	32	0	0	0	0	0

V → vitória; E → empate; D → Derrota; P → Não-entendeu

Tabela 3.19: Treinamento VI: Pontuação Contra Redes

3.9.7 TREINAMENTO VII

Ocorreram mais 200 partidas em 40 minutos, uma média de 5 partidas por minuto. Foi alterada a ordem das redes jogarem umas com as outras: Wall-e x C3PO, C3PO x R2D2, R2D2 x Número 5, Número 5 x Wall-e. Mantivemos a satisfação e insatisfação como no resultado anterior em quatro vezes para pontos e bloqueios de pontos e escolha de pino cheio, mantemos a regra em que a rede apenas tem uma chance de escolher um pino vago senão é contabilizado Não-entendimento. O único Não-entendimento que houve foi de C3PO para Wall-e e, apesar de ter perdido por Não-entendimento, não foi a que mais perdeu, isso mostra que a rede ainda não aprendeu 100% a regra do jogo, mas pode aprender jogando mais vezes. Em todos os confrontos ganhou mais vezes que iniciou as partidas. Mais uma vez Wall-e teve mais vitórias e fez mais pontos. Todas elas parecem estar bem equilibradas com pequena vantagem de Wall-e, que foi a que demorou mais a aprender a jogar e parece que é a que melhor aprendeu, talvez os pesos sinápticos tenham sidos corretamentes corrigidos, o resultado deste treinamento pode ser visto nas Tabelas 3.20 e 3.21.

	Pontos	Não-entendeu	Vitórias	Empates	Derrotas
R2D2	350	0	39	16	45
C3PO	362	1	41	16	43
Wall-e	393	0	46	13	41
Número 5	365	0	45	13	42

Tabela 3.20: Treinamento VII: Pontuação Por Rede

	R2D2				C3PO				Wall-e				Número 5			
	V	E	D	P	V	E	D	P	V	E	D	P	V	E	D	P
R2D2	0	0	0	0	13	11	26	0	0	0	0	0	26	5	19	0
C3PO	26	11	13	0	0	0	0	0	15	5	30	1	0	0	0	0
Wall-e	0	0	0	0	30	5	15	0	0	0	0	0	16	8	26	0
Número 5	19	5	26	0	0	0	0	0	26	8	16	0	0	0	0	0

V → vitória; E → empate; D → Derrota; P → Não-entendeu

Tabela 3.21: Treinamento VII: Pontuação Contra Redes

4 CONCLUSÃO

Este trabalho proporcionou o conhecimento sobre o que são as redes neurais artificiais, como elas funcionam, como aprendem e como fazer com que elas tomem decisões em um programa. O estudo iniciou-se a partir do estudo das características das redes. Em seguida, foi estudada sua estrutura e funcionamento. Tais redes são mais complexas do que parecem, possibilitando um grande grau de customização. Isto favoreceu a compreensão do porquê de existir uma vasta lista de modelos de redes neurais, cada modelo servindo a um ou mais propósitos, mesmo que um propósito pudesse ser servido por mais de um modelo de rede. Contudo, a tomada de decisão foi uma tarefa complexa. Durante o estudo, foi percebida a importância do treinamento para que as redes possam efetuar as operações que pretendemos que elas executem, existem também redes que podem aprender sozinhas tentando classificar as diferenças em seu ambiente.

Após esta tarefa de compreensão, foi decidido o desenvolvimento de um programa de computador para testar o seu funcionamento. Foi verificado que existem outras formas de se implementar redes neurais como alternativas ao desenvolvimento de software: implementações em circuitos eletrônicos e em cristais fotorefratários. Cada implementação com suas vantagens e desvantagens de acordo com a aplicação. Estudando como treinar redes para executar funções relativas ao programa escolhido, foi conhecido diversos algoritmos de aprendizagem, os quais são específicos para alguns modelos de redes. Há também a possibilidade de se otimizar o treinamento com a utilização de um processo de evolução da rede baseado em algoritmos genéticos.

A aplicação trata-se do Jogo da Velha 3D em Pinos implementada em software. Tendo em vista que o treinamento foi efetuado ao ocorrerem as jogadas durante os jogos e que este treinamento necessitou de muito tempo, foi decidido criar outras 3 redes. O treinamento entre as 4 redes neurais em confrontos individuais (1x1) possibilitou a diminuição do tempo, no sentido que não foi necessário criar situações de jogos. A criação do Juiz que sabe as regras favoreceu o aprendizado por que as redes poderiam basear-se em suas próprias jogadas para aprender. Assim, o processo de aprendizagem tornou-se bem mais rápido e independente de oponente

humano. Vale ressaltar que a qualidade das jogadas basei-se na decisão do juiz.

Foram efetuados 7 treinos, em um total de 35 horas e 10 minutos. sendo contabilizados 9500 confrontos. Ao final do treinamento, todas as redes aprenderam as regras do jogo, como pode ser visto na Tabela 3.20 observando-se a quantidade de Não-entendimentos. Contudo, como elas foram treinadas pelo programa juiz, aprenderam a jogar como o juiz joga, e o juiz não é preparado para jogar considerando armadilhas, que um humano pode fazer.

Este trabalho poderá servir como estudo introdutório para outros alunos que almejam estudar a área de redes neurais artificiais, sendo este não definitivo, por sabermos que os conhecimentos aplicados à informática e especificamente às redes neurais artificiais continuam evoluindo. Assim com o passar do tempo mais novidades e soluções serão desenvolvidas neste campo. Este trabalho também possibilita uma introdução nos estudos e aplicações na área de jogos de entreterimento e educativos que utilizam redes neurais artificiais.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, J. et al. Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review*, v. 84, n. 5, p. 413–451, 1977.

CARPENTER, G.; GROSSBERG, S. A neural theory of circadian rhythms: The gated pacemaker. *Biological Cybernetics*, Springer, v. 48, n. 1, p. 35–59, 1983.

CHANGEUX, J. et al. P. & Danchin, Selective stabilisation of developing synapses as a mechanism for specification of neuronal networks. *Nature*, v. 264, p. 705–12, 1976.

EGGERMONT, J. *The correlative brain: Theory and experiment in neural interaction*. [S.l.]: Springer-Verlag, 1990.

FAGGIN, F.; MEAD, C. Vlsi implementation of neural networks. Academic Press Professional, Inc., San Diego, CA, USA, p. 275–292, 1990.

FUKUSHIMA, K. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, Springer, v. 20, n. 3, p. 121–136, 1975.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, Springer, v. 36, n. 4, p. 193–202, 1980.

FUKUSHIMA, K. Neocognitron: A neural network for visual pattern recognition. *IEEE Computer*, v. 21, p. 65–75, 1988.

FUKUSHIMA, K.; MIYAKE, S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, Elsevier, v. 15, n. 6, p. 455–469, 1982.

FUKUSHIMA, K.; MIYAKE, S.; TAKAYUKE, I. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, v. 13, n. 5, p. 826–834, 1983.

GROSSBERG, S. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological cybernetics*, Springer, v. 23, n. 3, p. 121–134, 1976.

HAYKIN, S. *Redes neurais: princípios e prática*. Segunda. [S.l.]: Porto Alegre: Bookman, 2001. Trad. Paulo Martins Engel.

HEBB, D. The organization of behavior. New York. *Wiley. Hilgard, ER, & Marquis, DG (1940). Conditioning and learning. New\ brk: Appleton-Century-Crofts. Hoyle, G.(1979). Instrumental conditioning of the leg lift in the locust. Neuroscience Research Program Bulletin*, v. 17, p. 577–586, 1949.

HEBB, D. O. *The Organization of Behavior*. [S.l.]: John Wiley & Sons, 1949.

HINTON, G.; SEJNOWSKI, T. Learning and relearning in Boltzmann machines. *MIT Press, Cambridge, Mass.*, v. 1, p. 283–317, 1986.

HINTON, G.; SEJNOWSKI, T.; ACKLEY, D. Boltzmann machines: Constraint satisfaction networks that learn. *Cognitive Science*, v. 9, p. 147–169, 1984.

HOPFIELD, J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 79, n. 8, p. 2554, 1982.

KOHONEN, T. Automatic formation of topological maps of patterns in a self-organizing system. In: *Proceedings of the 2nd Scandinavian Conference on Image Analysis*. [S.l.: s.n.], 1981. p. 214–220.

KOSKO, B. Adaptive bidirectional associative memories. *Applied optics*, OSA, v. 26, n. 23, p. 4947–4960, 1987.

LOESCH, S. T. S. C. *Redes neurais artificiais: fundamentos e modelos*. [S.l.]: Blumenau: Ed. Da FURB, 1996.

MCCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, Springer, v. 5, n. 4, p. 115–133, 1943.

MEAD, C. *Analog VLSI and Neural Systems*. [S.l.]: Addison Wesley Publishing Company, 1996.

MEAD, C. *Analog VLSI and Neural Systems*. [S.l.]: Addison Wesley Publishing Company, 1996.

PAO, Y. Adaptive pattern recognition and neural networks. *Reading, MA*, 1989.

PINEDA, F. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, APS, v. 59, n. 19, p. 2229–2232, 1987.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, v. 65, n. 6, p. 386–408, 1958.

RUMELHART, D.; HINTON, G.; WILLIAMS, R. Learning Internal Representations by Error Propagation, D. Rumelhart, J. McClelland,(Eds), Parallel Distributed Processing: Exploration in the Micro-Structure of Cognition, volume 1. *Foundations MIT-Press*, p. 318–362, 1986.

SHEPHERD, G.; KOCH, C. Introduction to synaptic circuits. *The synaptic organization of the brain*, v. 0, p. 3–31, 1990.

STENT, G. A physiological mechanism for Hebb's postulate of learning. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 70, n. 4, p. 997, 1973.

SZU, H. Fast simulated annealing. In: *AIP conference proceedings*. [S.l.: s.n.], 1986. v. 151, p. 420.

TAFNER, M.; XEREZ, M. de. Filho, IWR (1995). *Redes Neurais Artificiais: Introdução e Princípios de Neurocomputação*, 1995.

TANK, D.; HOPFIELD, J. Neural computation by concentrating information in time. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 84, n. 7, p. 1896, 1987.

WERBOS, P. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Harvard University, 1974.

ZUBEN, F. V. Computação Evolutiva: uma abordagem pragmática. *Tutorial: Notas de Aula da disciplina IA707, Faculdade de Engenharia Elétrica e de Computação-Universidade Estadual de Campinas*, 2000.