CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

CAMPUS IV – ARAXÁ

ENGENHARIA DE AUTOMAÇÃO INDUSTRIAL

**Estudo comparativo entre Redes Neurais Artificiais e Algoritmos Genéticos aplicados na identificação de sistemas**

Maicon Fernando Dias Torres

Araxá

2014Maicon Fernando Dias Torres

**ESTUDO COMPARATIVO ENTRE REDES NEURAIS ARTIFICIAIS E ALGORITMOS GENÉTICOS APLICADOS NA IDENTIFICAÇÃO DE SISTEMAS**

Trabalho de conclusão de curso apresentado ao Curso de Engenharia de Automação Industrial, como requisito parcial para obtenção do título de Engenheiro de Automação Industrial.

Araxá

2014

**Dedico esse trabalho à minha família,**

 **aos meus professores**

**e aos meus colegas**

**que tornaram tudo isso possível.**

**AGRADECIMENTO**

Agradeço primeiramente a Deus por mais esse sonho concretizado, ao apoio psicológico da minha família e aos meus amigos que me incentivaram e ajudaram durante toda minha vida escolar. Agradeço aos meus professores por contribuir com seus conhecimentos para alcançar a minha formação, principalmente ao meu professor orientador Leandro Resende Mattioli por me guiar no desenvolvimento do trabalho, respeitando minhas ideias e contribuindo para solução de minhas dúvidas.

Muito Obrigado.

"A maior invenção do mundo não é a minha tecnologia! É a morte! pois através dela, o velho sempre dará lugar para o novo!"

Steve Jobs

**RESUMO**

A identificação de sistema é o ramo do estudo de controle que busca determinar o modelo que represente uma planta, visando realizar testes e simulações sem influenciar nos resultados físicos do sistema. Atualmente tem se optado por métodos baseados na natureza para solucionar as limitações da identificação matemática. Dentre os métodos mais utilizados e abordados nesse trabalho temos os Algoritmos genéticos e Redes Neurais Artificiais. Um problema acadêmico encontrado atualmente é a utilização “aleatória” do melhor método, há muitos estudos sobre cada um, individualmente ou em conjunto (híbrido), mas não existe um comparativo que determine as semelhanças e diferenças entre eles. O trabalho proposto pretende verificar a real capacidade de identificação desses métodos, e quais são suas limitações.

**Palavras chaves:**

Identificação, Rede Neural Artificial, Algoritmo Genético, identificação de sistemas.

**ABSTRACT**

The identification system is the control arm of the study that seeks to determine the model which represents a plan aiming at performing tests and simulations without influencing the results of the physical system. Currently has opted for nature-based methods to solve the limitations of the mathematical identification. Among the most discussed and methods used in this study have the genetic algorithms and neural networks. An academic problem encountered today is the "random" the best method, used for many studies about each, individually or in combination (hybrid), but there is no comparison to determine the similarities and differences between them. The proposed work aims to verify the actual ability to identify these methods, and what are its limitations.

**Key words:**

Identification, Artificial Neural Network, Genetic Algorithm, System Identification.

**LISTA DE FIGURAS**

[Figura 1 - Rede Perceptron 16](#_Toc403507451)

[Figura 2 - Neurônio biológico 17](#_Toc403507452)

[Figura 3 - Tipos de funções de limiar: (a) Degrau. (b) linear. (c) Sigmóide. (d) Tangente hiperbólica. 21](#_Toc403507453)

[Figura 4 - Exemplo de plasticidade em Redes neurais 22](#_Toc403507454)

[Figura 5 - MLP com uma camada oculta 24](#_Toc403507455)

[Figura 6 - Ilustração da retro-propagação do erro nas camadas da RNA 26](#_Toc403507456)

[Figura 7 - Fluxograma do algoritmo genético 29](#_Toc403507457)

[Figura 8 - Seleção por roleta 32](#_Toc403507458)

[Figura 9 - Seleção por torneio 32](#_Toc403507459)

[Figura 10 - Dizimação, exclusão dos valores de baixa aptidão (branco). 33](#_Toc403507460)

[Figura 11Cruzamento em um único ponto 34](#_Toc403507461)

[Figura 12 Cruzamento em ponto duplo 34](#_Toc403507462)

[Figura 13 Exemplo de cruzamento, troca de genes entre os pais, através de máscara. 35](#_Toc403507463)

[Figura 16 Sinal de entrada PRBS, usado durante o treinamento. 48](#_Toc403507464)

[Figura 17 Sinal de entrada Degrau unitário, usado na validação dos métodos. 49](#_Toc403507465)

[Figura 18 Gráfico representando a relação entre o fator de aprendizado, momento e o EMQ. 49](#_Toc403507466)

[Figura 19 Identificação do sistema para 50 épocas de treinamento 51](#_Toc403507467)

[Figura 20 Identificação do sistema para 100 épocas de treinamento 51](#_Toc403507468)

[Figura 21Identificação do sistema para 250 épocas de treinamento 52](file:///C%3A%5CUsers%5CMaicon%5CDownloads%5CTCCMaiconRevMattioliMod1.docx#_Toc403507469)

[Figura 22 Sistema real (vermelho) e sistema identificado pela AG (azul) 52](#_Toc403507470)

[Figura 23 Erro em função do número de épocas de treinamento 53](#_Toc403507471)

[Figura 24 Identificação por AG, geração 50. 53](#_Toc403507472)

[Figura 25 Identificação por AG, geração 100. 54](#_Toc403507473)

[Figura 26 Identificação por AG, geração 250. 54](#_Toc403507474)

[Figura 27 Identificação por AG, geração 500. 55](#_Toc403507475)

**LISTA DE TABELAS**

[Tabela 1 Exemplo de seleção por Roleta 31](#_Toc405969598)

[Tabela 2 Obtenção dos valores do sistema por recursividade do sistema discreto 42](#_Toc405969599)

[Tabela 3 - Comparação dos métodos. 55](#_Toc405969600)

**LISTA DE ABREVIATURAS**

RNA: Redes Neurais Artificiais.

AG: Algoritmo Genético.

PRBS: Sequência binomial pseudo-randomica.

Sumário

[1 Introdução 12](#_Toc405970404)

[2 Redes Neurais Artificiais 15](#_Toc405970405)

[2.1 Breve Histórico 15](#_Toc405970406)

[2.2 Sistema Nervoso Biológico 17](#_Toc405970407)

[2.3 Características principais 19](#_Toc405970408)

[2.3.1 Padrões 19](#_Toc405970409)

[2.3.2 Funções de limiar 20](#_Toc405970410)

[2.3.3 Conexões 21](#_Toc405970411)

[2.3.4 Elemento de processamento 21](#_Toc405970412)

[2.3.5 Plasticidade 22](#_Toc405970413)

[2.4 Aprendizado Supervisionado e não supervisionado 23](#_Toc405970414)

[2.5 Perceptron de Múltiplas Camadas (MLP) 23](#_Toc405970415)

[2.5.1 Treinamento e Algoritmo de Retro-Propagação 25](#_Toc405970416)

[3 Algoritmos Genéticos 28](#_Toc405970417)

[3.1 Função de aptidão 30](#_Toc405970418)

[3.2 O método de seleção 31](#_Toc405970419)

[3.3 Método de Cruzamento 33](#_Toc405970420)

[3.4 Método de mutação 35](#_Toc405970421)

[4 Identificação de Sistemas 37](#_Toc405970422)

[4.1 Sistemas caixa-preta, caixa-cinza e caixa-branca. 37](#_Toc405970423)

[5 Aplicação prática 39](#_Toc405970424)

[5.1 PRBS 39](#_Toc405970425)

[6 Analise dos métodos estudados 47](#_Toc405970426)

[6.1 Sistema 47](#_Toc405970427)

[7 Conclusão 57](#_Toc405970428)

Referências........................................................................................................59

# Introdução

Desde os primórdios da civilização o homem tem buscado pelo conhecimento e controle de tudo a sua volta. Com o advento da revolução industrial surgiram máquinas guiadas por controle humano, com o passar dos tempos os sistemas foram computadorizados, reduzindo a influência do homem no processo. Hoje em dia, o volume de informação dobra a cada quatro anos e num futuro próximo passará a dobrar a cada dois anos (Cerqueira et al.).

Os últimos tempos foram caracterizados pela tendência em buscar inspiração na natureza (TANOMURU, 1995), onde se procura cada vez mais igualar os conhecimentos de seres vivos aos sistemas de controle de processos, utilizando, dentre outros, Redes Neurais e Algoritmos Genéticos.

A identificação de sistemas é o método desenvolvido para representar as características físicas dos sistemas, sob a forma matemática. A capacidade de se ter um modelo matemático de um sistema permite vários estudos, tais como detecção de falhas, controle de processos, análise de comportamento complexo, entre outros (ROCHA, 2006). Além disso, a aplicação de testes no modelo matemático não causa danos à empresa, à planta e nem aos funcionários, permitindo que a planta fique parada o menor tempo possível, com o menor prejuízo possível. O campo de sistemas dinâmicos está conectado à identificação, previsão e controle de problemas onde as variáveis se interagem ao mesmo tempo (Ssu-Hsin, 1996).

O maior problema na identificação de sistemas está em encontrar a estrutura que representa melhor o modelo (SJÖBERG et. al, 1995). Atualmente muitos softwares são capazes de identificar um sistema através da aplicação de sinais PRBS e analises das saídas obtidas. Mas os softwares são limitados quando o sistema tem um comportamento incomum, uma vez que as maiorias dos sistemas atuais são complexos, multivariáveis ou não lineares, fica claro que os métodos matemáticos convencionais não conseguem obter com precisão a identificação do sistema nesses casos.

A solução encontrada são os métodos determinísticos, que consistem em obter aproximações probabilísticas ou estocásticas para encontrar informações do sistema, analisando apenas o comportamento das saídas e entradas. Normalmente são amplamente usados para identificação de sistemas dinâmicos, ou seja, sem que a planta tenha que parar suas funções para ser identificada, e em controles adaptativos com incerteza paramétrica.

As redes neurais artificiais (RNAs) possuem a capacidade de imitar o comportamento do cérebro para identificar e classificar padrões. Trata-se de uma importante técnica capaz de resolver uma gama de problemas complexos (Vellasco, 2007). Os algoritmos envolvidos ajustam os ganhos internos de uma rede neural de acordo com a realimentação das saídas do sistema físico. Esse método não soluciona o problema no sentido estritamente matemático, mas constituem técnicas de simplificação que fornecem uma solução aproximada do problema (Jovanovic, 2008).

Algoritmos Genéticos (AGs) têm sido amplamente usados em muitas aplicações para produzir uma solução global ótima (Zibo et. al., 1995) na estimação de parâmetros. Portanto, os AGs podem selecionar os melhores modelos a partir de outros métodos de identificação, de origem matemática ou neural. Em (Toshiro et. al., 2000) vemos a identificação de um sistema não linear com AG, que apresentou resultados com alta taxa de acerto. Isso significa que um sistema AG pode ser tão eficiente quanto uma Rede Neural Artificial.

Há um grande número de artigos utilizando essas ferramentas para identificação de sistemas. No entanto, normalmente os estudos focam nesses algoritmos apenas como ferramentas para a resolução dos problemas. No meio acadêmico, há poucos estudos comparando a eficiência desses métodos na determinação dos parâmetros de sistemas, de modo que os pesquisadores têm escolhido simplesmente pela aplicação em projetos semelhantes, ou melhor, adequação ao projeto. Com isso, fica clara a necessidade da elaboração deste estudo para enriquecer os conhecimentos dos futuros estudos na área de identificação de sistemas utilizando Redes Neurais Artificiais e Algoritmos Genéticos.

Estudos anteriores analisam cada método de forma individual, e não de forma comparativa, assim é possível que os resultados obtidos nesse trabalho possam servir de referência para futuros projetos de identificação de sistema usando cada método separado ou em conjunto (híbrido).

Nesse trabalho de TCC apresentamos um estudo comparativo da utilização de RNAs e AGs para identificação de um sistema caixa preta. Os algoritmos são programados na linguagem Delphi, devido à maior experiência do autor do trabalho.

O presente trabalho tem como objetivo principal suprir e contribuir com a necessidade acadêmica de estudos que referenciam o uso de Redes Neurais e Algoritmos Genéticos e comparar a aplicação desses métodos na identificação de sistemas caixa preta, além de determinar qual é vantagem de se utilizar esses algoritmos.

O algoritmo da rede neural será configurando para identificar um sistema de primeira ordem conhecido, possibilitando dessa forma, determinar se a funcionalidade da RNA para esse fim pode ser alcançada. Em seguida será desenvolvido e configurado um AG para identificar o mesmo sistema de primeira ordem, com o mesmo objetivo da RNA. Com ambos os métodos desenvolvidos e funcionais, aplicaremos aos mesmos a função principal do nosso problema, uma função discreta de segunda ordem, obtendo para cada método o erro médio quadrático com relação ao sistema original. De posse desses dados, conseguiremos plotar, analisar e classificar os algoritmos quanto à dificuldade do sistema a ser identificado e o erro médio quadrático, para cada caso.

# Redes Neurais Artificiais

## Breve Histórico

 Há algumas décadas atrás, pensar em uma máquina capaz de aprender algo era totalmente absurdo. O cérebro conseguia fazer tarefas complexas em milésimos de segundos enquanto o computador mais avançado demorava dias para resolvê-los. Mas foi em 1943 que o neurofisiologista McCulloch e o matemático Water Pitts da universidade de Illinois apresentaram, pela primeira vez ao meio acadêmico, um circuito inspirado em células nervosas. O dispositivo era uma unidade binária composta por resistores e amplificadores com a capacidade de executar diversas funções lógicas.

Em 1948, o matemático e filósofo Norbert Wiener criou a palavra cibernética para descrever o controle e comunicação nos organismos vivos e máquinas. No final da década de quarenta, o psicologista Donald O. Hebb apresentou no livro "The Organization of Behavior" a ideia de pesos sinápticos como forma de armazenamento de informações em uma rede neural. No trabalho dele quatro temas foram citados (Eberhart & Dobbins, 1990):

1. Numa rede neural a informação é armazenada nos pesos;
2. O coeficiente de aprendizagem é proporcional ao produto dos valores de ativação do neurônio;
3. Os pesos são simétricos;
4. Quando ocorre o aprendizado os pesos são alterados.

Em 1957 o coronel do laboratório da aeronáutica em Nova York criou um algoritmo de classificação linear com classificação supervisionada. O algoritmo combinava um vetor de características com pesos, que são ajustados durante a fase de treinamento. Esse algoritmo ficou conhecido como Perceptron. Cinco anos depois o professor de Engenharia Elétrica de Stanford, Bernard Widrow e o doutorando Ted Hoff inventaram o filtro de mínimos quadrados (LMS) que consistia em um filtro adaptativo. Widrow e Hoff propuseram uma modificação no método de treinamento do Perceptron, mais precisamente ajustar os pesos de acordo com soma da faixa de pesos das entradas. Widrow e Hoff denominaram esse algoritmo como ADALINE (Adaptative Linear Element).

Durante a década de 60, as pesquisas em redes neurais sofram reduções significativas, devido à carência de resultados teóricos como incentivo ao meio acadêmico. Além disso, os pesquisadores criavam grandes expectativas sobre os trabalhos, mas não alcançavam os resultados à altura.

Os estudos ganharam força após a publicação do livro "Perceptrons" de M. L. Minsky e S. A. Papert, em 1969, que trazia conceitos de matemática moderna como analise das capacidades adaptativas e computacionais dos neurônios. Os autores destacaram a limitação do perceptron em executar outras operações elementares como XOR, e não apenas operações booleanas básicas como AND e OR.



Figura 1 - Rede Perceptron

Fonte: <http://rodrigoregis.blogspot.com.br/2012/08/o-que-e-uma-rede-neural-artificial.html>, acessado em 17/09/2014.

As pesquisas sobre redes neurais alavancaram com a utilização do algoritmo de retro-propagação (backpropagation) para ajustar os parâmetros em redes de múltiplas camadas, apresentada em 1986 no livro Parallel Distributed Processing, editado por J. L. Mcclelland & D. E. Rumelhart, o que chamou a atenção de pesquisadores das mais diversas áreas. Atualmente as redes neurais artificiais são aplicadas nas mais diversas áreas desde a robótica até o setor de finanças.

## Sistema Nervoso Biológico

O cérebro é o órgão central do sistema nervoso, formado por 10 bilhões de neurônios, interconectados entre si através das sinapses. Eles formam uma grande rede denominada Rede Neural. O sistema nervoso é responsável pelo recebimento e controle das respostas musculares e glandulares, através estímulos externos e internos, para garantir a integração do organismo com o meio ambiente. A união das células nervosas em circuitos neurais possibilita que o organismo produza respostas estereótipos para comportamentos fixos e invariantes, como os reflexos, ou em comportamentos variáveis em graus de intensidade, como por exemplo, dor e calor.



Figura 2 - Neurônio biológico.

 Fonte: <http://www.portalsaofrancisco.com.br/alfa/corpo-humano-sistema-nervoso/sistema-nervoso-1.php>, acessado em 18/06/2014.

Um neurônio em repouso apresenta uma carga positiva interna da membrana e negativa na parte externa; nessa condição diz-se que a célula está polarizada.

Para manter essa diferença de potencial a célula utiliza um processo denominado bomba de sódio-potássio, que consiste na troca involuntária de íons de sódio e potássio através da membrana. Qualquer estimulo químico, mecânico ou elétrico, aplicado no neurônio ocasiona a alteração da permeabilidade da membrana, permitindo a entrada de sódio e a saída de potássio na célula. Esse evento provoca inversão de potencial elétrico e a consequente despolarização da célula. A diferença de cargas geradas na membrana se propaga pelo neurônio, em uma reação em cadeia. O axônio transmite esse impulso elétrico a outros neurônios por meio das sinapses.

Após a emissão do impulso, a célula despolarizada usa a bomba de sódio-potássio para regenerar sua carga positiva. Ao mesmo tempo, dendritos de outros neurônios captam o impulso emitido, e o conduz pelo soma. Esse neurônio iniciará a mesma reação de despolarização e estimulará outros neurônios, mas se o potencial não atingir um valor mínimo para induzir a despolarização da célula, o impulso não é retransmitido, inibindo uma reação do sistema.

 A intensidade do pulso necessário para a retransmissão do impulso é denominado estimulo limiar. Abaixo desse valor as alterações serão apenas locais e cessam antes chegaram às sinapses.

Nos seres humanos essa habilidade dos neurônios é denominada sensibilidade ou a tolerância a sensações. Algumas pessoas fotossensíveis não toleram ou sofrem grande desconforto, ao andar durante o dia à luz do sol, há relatos de pessoas com esse problema afirmam que têm a sensação de pequenos choques quando expostos ao sol, em contra partida algumas pessoas podem expor-se a temperaturas abaixo de zero e não ter qualquer reação ao frio.

Os estímulos emitidos por cada neurônio têm sempre a mesma intensidade binária, máxima ou nenhuma. A intensidade do estimulo recebido pode alterar de acordo com a combinação de estímulos de outros neurônios, em outras palavras, quanto maior o número de células despolarizadas e quanto maior for a frequência de impulsos, maior será o estimulo recebido, consequentemente maior será probabilidade do impulso ser transmitido aos neurônios seguintes.

## Características principais

Os principais componentes físicos de uma rede neural artificial são as conexões e os elementos de processamento. Dentre os componentes não físicos temos os padrões e funções. Os padrões são variações dos dados da entrada que correspondem à mesma saída. O treinamento e o reconhecimento de padrões compõem modelos matemáticos denominados funções.

### Padrões

Os ganhos são os responsáveis por armazenar o aprendizado da Rede Neural, mas toda informação está nos dados. Como no mundo real nem sempre as informações estão no mesmo formato das entradas da rede (matriz de números), é necessário encontrar um padrão que corresponda a todas as variações das entradas em relação com a saída. Considerarmos a identificação de dígitos em uma placa de carro. Se a foto foi tirada de perto teríamos dígitos grandes, se tirarmos de longe teriam dígitos pequenos, outros fatores poderiam criar variações dos dígitos como inclinação, ruídos, danos à placa, cor etc. Mas o Digito “1” sempre corresponderá à saída “um” não importando o seu tamanho ou inclinação. Portanto nessa condição é necessário um pré-processamento dos dados e a conversão da informação em uma matriz de números com a mínima interferência dos fatores citados; isso simplifica e aumenta a taxa de sucesso da rede.

### Funções de limiar

A principal função de uma rede neural é a função de limiar, comumente chamada de função de ativação. Cada neurônio faz o somatório das entradas ponderadas pelos ganhos para encontrar um valor de impulso. Esse é então aplicado à função de limiar, na qual será determina o formato e intensidade do impulso transmitido aos neurônios seguintes. Dentre as principais funções de limiar estão a linear, o degrau, a rampa, a sigmóide e a gaussiana.

Na função linear a saída do neurônio é proporcional ao impulso do somatório. A função degrau corresponde à comutação brusca da saída ao atingir o limiar (tudo ou nada). A função rampa é a combinação da função degrau com a linear: a saída é proporcional durante a transição e constante fora dos limites. A sigmóide é a versão continua da rampa, a diferença está no fato das transições serem graduais e não lineares entre os dois estados. A gaussiana é amplamente utilizada em estatística; normalmente é utilizada quando há um ponto médio em X e uma variação nesse mesmo ponto.



Figura 3 - Tipos de funções de limiar: (a) Degrau. (b) linear. (c) Tangente hiperbólica. (d) Sigmóide.

Fonte: acervo do autor.

### Conexões

As conexões são responsáveis por ligar um nó a outro, direcionando o sentido de propagação da informação. Cada conexão tem um ganho ajustado durante a fase de aprendizado. Quando esse ganho é positivo a conexão é excitatória, ou seja, intensifica o impulso. No caso de negativo a ação é inibitória, reduzindo a intensidade. O ganho também pode ser nulo, nesse caso a conexão inibe todo sinal que seja transmitido por ela, é como se não existisse a conexão.

### Elemento de processamento

O neurônio ou neuronodo é o elemento de processamento da Rede Neural. Cada neurônio capta as informações nas entradas, pondera com pesos e produz uma única saída, que é enviada a todos os neurônios das camadas seguintes. O fato dos neurônios necessitarem apenas de informações locais (pesos e entradas) e as saídas serem sempre para os neurônios posteriores, garante uma vantagem sobre os computadores convencionais: O processamento em paralelo.

O processamento distribuído ou paralelo permite que cada elemento de uma camada trabalhe apenas com parte da informação, os resultados dos cálculos são enviados à próxima camada, o que permite que cada elemento de processamento seja mais simples e o conjunto produza respostas mais rápidas. Isso contradiz com a abordagem dos computadores convencionais, que, por mais rápidos que sejam, executam um cálculo de cada vez.

### Plasticidade

A plasticidade neural ou generalização é a capacidade da rede neural gerar uma saída, para entradas que não estavam na fase de aprendizado. Essa característica é possível devido aos padrões das entradas para as entradas aprendidas.

Além disso, tal comportamento diferencia efetivamente as redes neurais das funções. Tanto as funções quanto as redes neurais recebem um ou mais parâmetros, ou entrada de dados, e retorna um resultado, uma saída. No entanto, enquanto uma função deve conter explicitamente um valor de retorno para qualquer entrada, as redes podem receber apenas intervalos das informações durante a fase de treinamento e, por indução, gerar as saídas para entradas desconhecidas.

Considere uma rede neural treinada para reconhecer duas cores, preto e branco. As entradas podem assumir 255 valores correspondentes a diferentes tons de cinza. Abaixo do nível 127 o elemento é considerado branco; acima deste mesmo limite o pixel é visto como preto. Na fase de treinamento, apenas os tons pares são usados. Depois que a rede for treinada insere na entrada os tons impares, o resultado da rede será correto para quase todos os valores.



Figura 4 - Exemplo de plasticidade em Redes neurais.

Fonte: acervo do autor.

## Aprendizado Supervisionado e não supervisionado

A base do aprendizado supervisionado é o aprendizado associativo, ou seja, cada entrada está associada a uma saída. Durante a fase de treinamento os dados são inseridos na entrada da rede, os neurônios processam os dados e geram uma saída, que é então comparada com a saída desejada. A diferença entre a saída real e a desejada é o erro, utilizado para ajustar os pesos sinápticos da rede neural. Os algoritmos de treinamento minimizam o erro até que esse seja insignificante; a partir desse momento considera-se que a rede aprendeu o padrão ensinado.

Quando a rede é baseada no aprendizado não associativo não há uma associação das entradas com as saídas. Nesse caso a repetição de um estímulo fornece a oportunidade para aprender suas propriedades. Esse modelo normalmente é utilizado em redes neurais não supervisionadas. Nesse tipo de rede não há saída desejada; durante a fase de treinamento os dados inseridos nas entradas são auto-organizados, arbitrariamente, em categorias de acordo com os padrões. Caso uma entrada não se adeque a nenhuma categoria, um nova é criada. Esse tipo de rede é amplamente utilizado em sistemas de classificação.

## Perceptron de Múltiplas Camadas (MLP)

A MLP é a expansão da rede Perceptron simples, com a adição de camadas intermediarias entre a entrada e a saída. Essas camadas internas permitem à MLP solucionar problemas complexos, difíceis e não lineares, além de criar mais interações das camadas de entrada, impulsionando sua implantação na grande maioria dos problemas de alta complexidade.

As Perceptron de múltiplas camadas normalmente utilizam o algoritmo de retro-propagação do erro para calibrar os pesos de suas camadas. Este algoritmo é baseado na regra de aprendizado de correção de erro (Haykin). A retropropagação do erro consiste, basicamente, em dois passos pelas camadas. O primeiro é o passo direto, nesse modo um sinal é aplicado nas entradas, o sinal se propaga para saída por meio dos neurônios internos ou ocultos. O sinal obtido na saída é comparado com o resultado esperado para aquele sinal da entrada. O erro encontrado é utilizado na segunda fase do aprendizado, a retropropagação do erro. Nessa fase a rede calibra os valores dos pesos de cada camada de acordo com o valor do erro, partindo da camada de saída, passando pelas ocultas e finalizando na camada de entrada. Após o treinamento, a MLP só trabalha no primeiro modo, propagação direta.



Figura 5 - MLP com uma camada oculta.

Fonte: <http://www.lncc.br/~labinfo/tutorialRN/frm1_introducao.htm>, acessado em 16/09/2014.

As MLPs trabalham com basicamente com dois tipos de sinais, o sinal erro e sinal de entrada.

* Sinal de entrada é um estímulo aplicado na entrada e se propaga, através dos neurônios, até a saída da rede neural. O valor do sinal função é calculado pela multiplicação dos pesos, ao passar pelas camadas.
* Sinal Erro é que origina na saída da rede e propaga até o início, na camada de entrada, passando por cada camada intermediária.

### Treinamento e Algoritmo de Retro-Propagação

Há centenas de algoritmos de treinamento para redes neurais, no entanto o mais difundido é o *backpropagation* (backward propagation of error), por envolver cálculos simples que realizam um gradiente descendente no espaço de conexões. O algoritmo é dividido em três fases: propagação “para frente” (feed forward) do sinal de entrada, propagação ”para trás" (backpropagation) do erro da saída, e ajuste dos pesos sinápticos.

A desvantagem do backpropagation está no tempo de treinamento, pois é muito longo para as aplicações práticas.



Figura 6 - Ilustração da retro-propagação do erro nas camadas da RNA.

Fonte: acervo do autor.

O aprendizado da Perceptron de múltiplas camadas provém do ajuste dos pesos de forma a convergir no erro nulo. Esse ajuste é realizado em passos de tamanho definidos. O valor desse passo mínimo é denominado taxa de aprendizado. Um valor alto da taxa de aprendizado pode significar aproximar do objetivo final mais rápido, mas também pode criar oscilações entre o ponto de aprendizado ideal. Por outro lado um valor muito pequeno pode tornar o treinamento muito lento, contudo pode convergir para o melhor conhecimento possível. A conclusão que se tira a respeito da taxa de aprendizado é que deve se achar um ponto de equilíbrio e que apenas com numerosos testes dessa gama de valores poderá atingir o valor ideal.

O processo de aprendizagem se repete por várias vezes, para cada par entrada e saída. Essa é a forma da rede ajustar seus ganhos das camadas não apenas para uma saída isolada, mas para a interação de todos os sinais de entrada. É dessa forma que a MLP consegue reagir a estímulos que não estavam na fase de aprendizado, mas que, por aproximação, correspondem a certa saída estimada. O treinamento se encerra segundo um critério de parada. O mais comum é a definição de um erro médio quadrático (EMQ) máximo. Quando o EMQ é suficientemente pequeno, considera-se a rede treinada e o algoritmo de treinamento é finalizado.

# Algoritmos Genéticos

No ano de 1958 o botânico Charles Darwin apresentou o conceito da seleção natural, para provar a teoria da evolução das espécies. O conceito defende a ideia de que a espécie que sobrevive não é a mais forte, e sim a que sofre o maior número de adaptações ao ambiente. Essa teoria é considerada uma das mais importantes da modernidade.

Posteriormente, as novas descobertas científicas da genética contribuíram para a criação do neo-darwinismo. Essa nova ideia segue os seguintes preceitos:

* Todos os indivíduos estão continuamente disputando os recursos do meio ambiente;
* Aqueles que possuem características específicas possuem maior chance de sobreviver (indivíduos mais adaptados ao meio ambiente);
* Indivíduos mais adaptados sobrevivem e reproduzem com maior probabilidade;
* A reprodução garante a propagação das características específicas para seus descendentes;
* A propagação de características positivas garante uma nova geração mais adaptada às condições do ambiente;
* Durante a fase de reprodução pode haver uma falha na transmissão dos genes, essa falha denomina mutação. Normalmente a mutação é prejudicial ao indivíduo, no entanto pode garantir aos recendentes características positivas que não pertence a nenhum pai, assim a natureza garante o maior número de combinações e possibilidades de características.

Os algoritmos genéticos foram criados baseando-se no princípio da seleção natural, visando solucionar problemas de otimização. Essa técnica é caracterizada pela robustez e pela facilidade de adaptação às mais diferentes condições.

O funcionamento do algoritmo é alcançado submetendo uma população de soluções para um problema ao processo de evolução. Esse processo é formado pelas sete etapas abaixo:

* Avaliação: Verificação da aptidão das soluções para definir a qualidade de cada solução;
* Seleção: Escolha de uma solução, baseando na aptidão;
* Cruzamento: Recombinação dos indivíduos selecionados para criação de descendentes;
* Mutação: Alteração de uma característica dos indivíduos;
* Atualização: Os descendentes são inseridos na população;
* Finalização: Verificação da aptidão das soluções na população, caso a aptidão ótima não seja alcançada, uma nova iteração de evolução é executada.



Figura 7 - Fluxograma do algoritmo genético.

Fonte: acervo do autor.

## Função de aptidão

A função de aptidão é a função que determina quão próximo da solução ideal está um dado indivíduo. Mesmo sem saber qual é a solução ótima para um problema, é possível determinar um ponto de convergência para a solução. Por exemplo, imagine um problema que procura as raízes de uma equação do segundo grau. É sabido que a substituição da incógnita X, pelo valor das raízes, deve ser zero, essa é a definição de raiz de equação. Se utilizarmos raízes com valores aleatórios, o resultado da equação será diferente de zero, quanto mais distantes dos valores corretos, mais distantes de zero a imagem da função estará. Dessa forma é possível identificar uma tendência das soluções, quanto menor o valor da imagem, melhores são as raízes (mais próximas das corretas).

$$f\left(X\right)=X^{2}+X+1=0$$

Normalmente, os valores da função de aptidão são limitados entre zero e um, para facilitar a análise e evitar a convergência para infinito. Como no exemplo anterior a equação de segundo grau tende a zero, para a solução ideal e para mais ou menos infinito para a pior solução, precisamos limitá-la no intervalo unitário, assim podemos utilizar a função abaixo como função de aptidão.

$$f\_{apt}=\frac{1}{1+\left|f\left(X\right)\right|}$$

$$ $$

A função de aptidão acima aplica o valor da equação na função módulo, isso permite limitar o intervalo de mais infinito até zero. Assim, para o maior erro possível, tendendo a infinito, a função de aptidão tenderia a zero. Da mesma forma, a melhor solução seria quando |f(x)| tende a zero o resultado 1. Logo, conseguimos qualificar a solução de um problema sem saber o valor da solução ideal, esse é o objetivo da função de aptidão.

## O método de seleção

Os algoritmos genéticos possuem diversos métodos de seleção. A escolha inadequada desse método pode causar atrasos na execução ou não alcançar o ótimo global. A seleção ocorre após o cálculo da aptidão dos elementos gerados. Através da aptidão é possível determinar quais os cromossomos possuem maior probabilidade de gerar descendentes que tendam a solução ótima. Aqueles que têm baixa probabilidade também podem gerar descendentes, mas com menor incentivo. A combinação dos cromossomos de baixa aptidão pode gerar um descendente de alta aptidão, devido a características que os demais da elite não possuem.

A técnica da Roleta é muito utilizada devido à simplicidade de execução. Todos os cromossomos recebem uma porcentagem da roleta proporcional à sua aptidão, ou seja, quanto maior for a aptidão de um cromossomo maior será sua a probabilidade de ser escolhido. Após definida a roleta, gera-se um número aleatório, dentro dos limites da mesma, deixando a sorte definir qual será o cromossomo selecionado. O processo se repete até que seja obtido um número par necessário para executar o processo de cruzamento e mutação.

Tabela 1 Exemplo de seleção por Roleta

|  |  |
| --- | --- |
| **Cromossomo** | **Valor de Aptidão** |
| 0001b | 1 |
| 0011b | 9 |
| 0100b | 16 |
| 0110b | 36 |

Abaixo vemos o espaço ocupado da roleta, ou seja, os pesos de cada cromossomo, de acordo com a aptidão. Observe que mesmo o individuo com a menor aptidão (0001b), tem uma pequena chance de ser selecionado, isso porque há uma pequena chance de haver um ótimo global, e os demais cromossomos estão perto de um ótimo local.

Figura 8 - Seleção por roleta.

Fonte: acervo do autor.

O torneio é um método parecido com a roleta. No entanto, o sorteio é igualitário, ou seja, todos os indivíduos possuem a mesma probabilidade de serem escolhidos, e só após o sorteio é analisada a aptidão dos cromossomos. Inicialmente o método escolhe aleatoriamente n cromossomos da população atual. Os escolhidos com maior aptidão são selecionados para uma população intermediária, os demais são reinseridos na população atual. O processo é repetido até que esteja completa a população intermediária.



Figura 9 - Seleção por torneio.

Fonte: acervo do autor.

A Dizimação é a técnica de seleção de cromossomos onde, inicialmente, a população é ordenada segundo seu valor de aptidão. Em seguida um número fixo de cromossomos de baixo valor de aptidão é removido da população. O processo se repete até que sobrem na população apenas aqueles que cruzarão para formar a nova população. Como mencionado no primeiro paragrafo, executar a exclusão dos indivíduos menos aptos pode ser ruim para as populações futuras, pois pode-se perder características positivas desses cromossomos. Esse fato é considerado como a principal desvantagem para essa técnica.



Figura 10 - Dizimação, exclusão dos valores de baixa aptidão (branco).

Fonte: acervo do autor.

## Método de Cruzamento

O cruzamento é a fase do Algoritmo genético que segue após a seleção. Assim como no cruzamento biológico, cada progenitor sede parte de suas características para criar um novo indivíduo. Normalmente essas características são positivas, e fazem com que o descendente seja mais apto que os progenitores. As formas de cruzamento mais conhecidas são as de ponto único, duplo e cruzamento de pontos aleatórios.

A ideia do ponto único é escolher um aleatoriamente um ponto de corte e a partir desse ponto combinar os dois cromossomos para originar dois novos indivíduos, mesclando consequentemente as características genéticas dos pais. O processo pode se repetir para outros pontos caso um par de genitores origine dois novos descendentes.



Figura 11Cruzamento em um único ponto.

 Fonte: Acervo do autor.

Outra forma de fazer o corte dos cromossomos é escolher dois pontos aleatórios, entre os quais os materiais genéticos são trocados. Esse método é conhecido como método do ponto duplo.



Figura 12 Cruzamento em ponto duplo.

 Fonte: acervo do autor.

O terceiro método apresentado é conhecido como pontos aleatórios, e consiste em utilizar uma máscara binária (contendo 0s e 1s) aleatória. Em seguida a rotina varre os alelos dos cromossomos segundo o valor do bit da máscara. Quando a mascara contiver o bit igual a 1, o descendente herda o alelo do pai 1; quando for 0, herda o alelo do pai 2. O descendente é formado combinando cada alelo herdado dos pais. O segundo descendente é obtido através da inversão da mascara e repetindo o processo.



Figura 13 Exemplo de cruzamento, troca de genes entre os pais, através de máscara.

Fonte: acervo do autor.

## Método de mutação

O processo de mutação consiste em alterar aleatoriamente algumas características dos indivíduos da população. Isso é de vital importância para alcançar a solução ótima para o problema, pois é possível que nenhum dos pais tenha, ou tenha em baixa porcentagem, alguma propriedade especial para acelerar o processo de busca da solução ideal. O principal fator que se deve ter em mente é que a mutação deve atingir apenas a uma pequena parte da população, para evitar a perda de características ou a intensificação de características negativas. Dentre as mutações mais utilizadas encontra-se a mutação aleatória e a mutação por trocas.

Na mutação aleatória é sorteado um alelo, que é alterado de acordo com alfabeto de valores válidos. Na mutação por troca, n pares de alelos são sorteados aleatoriamente e permutados entre si.

# Identificação de Sistemas

O controle de sistemas surgiu logo após a revolução industrial, onde foram criadas plantas com o objetivo de substituir ou auxiliar o trabalho humano.

Os primeiros sistemas eram simples quando comparados com os atuais, a grande maioria mecânicos e pneumáticos, ambos com controle manual.

Com o surgimento dos motores elétricos, vieram junto todos os dispositivos para o seu controle, como chaves fim-de-curso, contadores e disjuntores, tornando o sistema mais complexo. Após a descoberta dos semicondutores, os dispositivos passaram a ser cada vez menores e com o funcionamento mais oculto, o controle passou a ser mais software do que hardware.

## Sistemas caixa-preta, caixa-cinza e caixa-branca.

As plantas ou sistemas cujas características são desconhecidas são denominados sistemas caixa-preta. O termo caixa-preta refere-se à ideia de "estar no escuro", ou seja, não se consegue ver qualquer informação a respeito do sistema.

Na prática, sempre se tem o conhecimento de alguma característica da planta, mesmo que seja por intuição. O formato da resposta da planta é um parâmetro que pode ser reconhecido. Quando se tem certa experiência com controles, é possível determinar se uma planta é de primeira ordem ou de ordem superior, simplesmente verificando a estabilidade da resposta. Quando temos um sistema onde alguns parâmetros são conhecidos, atribui-se a denominação de sistema caixa-cinza, ou seja, é possível ver apenas algumas informações do sistema, mas não é o suficiente para representá-lo.

Vários métodos foram desenvolvidos com o objetivo de identificar os parâmetros ocultos de sistemas. O modelamento matemático é um método que combina elementos básicos, cujas fórmulas são conhecidas e que constituem o sistema, para encontrar a fórmula matemática global que representa todo ou parte do funcionamento do sistema.

O método das análises das frequências é uma forma de encontrar polos e zeros da função de transferência de uma planta. Os Polos e Zeros são as raízes do numerador e denominador, respectivamente, da função de transferência. A determinação desses parâmetros significa a obtenção da função que representa o sistema. O método inicia com a aplicação de um espectro de frequência nas entradas; as leituras das saídas são armazenadas e plotadas. O gráfico obtido representa o diagrama de fase do sistema, onde cada variação dos ângulos representa o valor de um polo ou zero. O maior problema desse método é que muitos sistemas não aceitam a aplicação de frequências nas suas entradas, ocasionando perda da estabilidade ou até mesmo danos físicos, como é o caso das válvulas, e pistões.

Outros métodos baseiam na análise da resposta da planta para entradas conhecidas. O método inicia com a aplicação de um sinal pseudorrandômico na entrada, e a aquisição dos dados da saída. Esses dados são processados para obter os parâmetros que ajustem um modelo básico conhecido. Existem vários modelos básicos para representar um sistema, mas a escolha do melhor só pode ser determinada através de testes e comparações.

Quando é possível representar um sistema e obter as mesmas respostas, mesmo sem o contato com o sistema físico, dizemos que o sistema foi identificado.

# Aplicação prática

A linguagem de programação Delphi foi utilizada na implementação dos algoritmos. Devido à experiência prévia, a versão XE2 do compilador foi escolhida.

O trabalho iniciou com a escolha de uma função de transferência de primeira ordem discreta, com o objetivo de provar que os algoritmos conseguem replicar a resposta da função escolhida apenas pela análise das entradas e saídas. Identificar um sistema de primeira ordem é um pré-requisito mínimo para qualquer método de identificação de sistema, portanto esse foi o teste primário dos algoritmos. A equação característica do sistema é:

$$Y\left[k\right]=a∙U\left[k\right]\pm b∙Y\left[k-1\right]$$

O segundo parâmetro principal para a identificação correta do sistema é o sinal aplicado nas entradas. Para que os algoritmos identifiquem a resposta da função para cada frequência de sinal da entrada, optamos pelo PRBS (*pseudorandom binary sequence*) que consiste em uma sequência binária randômica, mas periódica.

## PRBS

O algoritmo do PRBS gera uma sequência pseudorrandômica de bits baseadas no conceito de realimentação linear de registradores. A geração do sinal de PRBS pode ser representada pelo deslocamento de um registrador com o número de bits igual ao número de bits aleatórios do período, e em seguida aplicando a função XOR entre os bits e realimentado o registrador com o bit obtido. Esses bits são acumulados em uma sequência de bits que se repetem ao fim do período.

Para entender o funcionamento do gerador é necessário conhecer o conceito de registrador de deslocamento com realimentação linear (LFSR). Um registrador de deslocamento é uma sequência de flip-flops conectados em cascata, de maneira que a saída de um é a entrada do seguinte. Isso significa que o bit inserido na entrada do primeiro flip-flop (entrada do registrador) será deslocado para o próximo, o próximo transmitirá para o seguinte e assim sucessivamente.

Para a geração do sinal pseudorrandômico precisamos realimentar o registrador de deslocamento, por meio da aplicação da função XOR em dois bits internos do registrador, esses bits são determinados segundo a sequência necessária, a mais comum é a PRBS7, que aplica a XOR entre os bits 6 e 7.

Em linguagem de programação consideramos o um vetor de bits como um registrador e o deslocamento é a substituição dos valores das posições superiores pelos valores das inferiores. A última posição do vetor representa o valor do bit na sequência, e será aplicado na XOR juntamente com o bit “b-j-1”, onde o valor “j” representa qual polinômio será utilizado, a partir do tamanho da sequência “b”. O resultado da XOR é reinserido na posição zero (entrada do registrador). Abaixo temos o algoritmo utilizado no trabalho para geração do sinal.

**type**

 TData **=** **array** **of** double**;**

{

PRBS( N:número de elementos

 b:número de elementos sem repetição

 Amplitude: valor escalar do sinal binário);

Result: vetor de dados binários pseudorrandômicos

}

**function** PRBS**(**N**,** b**,** m**:**integer**;** Amplitude**:**double **=** 1**):** TData**;**

**var** tmp**:**double**;**

 j**,**i**,**k**:** Integer**;**

 x**:**TData**;**

 **function** \_Xor**(**a**,**b**:**Double**):** Double**;** // função OR EXCLUSIVO

 **begin**

 **if** **(**a **=** 1.0**)and(**b **=** 0.0**)**

 **or** **(**a **=** 0.0**)and(**b **=** 1.0**)** **then** Result**:=** 1.0

 **else**

 Result**:=** 0.0**;**

 **end;**

**var**

 buf**:string;**

**begin**

 SetLength**(**Result**,**N**);**

 SetLength**(**x**,**b**);**

 buf**:=**''**;**

 **for** i **:=** 0 **to** b**-**1 **do**

 **begin**

 x**[**i**]:=** 1.0**\***Random**(**2**)\***Amplitude**;**

 buf**:=** buf**+**FloatToStr**(**x**[**i**])+**','**;**

 **end;**

 **for** i **:=** 0 **to** N**-**1 **do** // inicializa o vetor

 Result**[**i**]:=** 0.0**;**

 **case** b **of** // o índice "j" tem valores

 5**:** j**:=** 2**;** // específicos para certos

 7**:** j**:=** 3**;** // valores de "b"

 9**:** j**:=** 4**;**

 10**:** j**:=** 3**;**

 11**:** j**:=** 2**;**

 **else** j**:=** 1**;** // para os demais j=1

 **end;**

 **for** i **:=** 0 **to** N**-**1 **do**

 **begin**

 Result**[**i**]:=** x**[**b**-**1**];** // aplica a XOR

 tmp**:=** \_Xor**(**Result**[**i**],**x**[**b**-**j**-**1**]);** // entre “X” e o “Result”

 **for** k **:=** length**(**x**)-**2 **downto** 0 **do** // desloca os bits

 x**[**k**+**1**]:=** x**[**k**];** // (shift register)

 x**[**0**]:=** tmp**;**

 **end;**

**end;**

Para criar o par, a entrada e a saída do sistema, um vetor bidimensional foi preenchido com função discreta do sistema simulado, pelo método recursivo, ou seja, aplicando as saídas anteriores para gerar a saída atual.

$$U\_{k}=PRBS(k)$$

$$Y\_{k}=aU\_{k}-bY\_{k-1}$$

A tabela abaixo é um exemplo da aplicação de um sinal degrau na função de transferência abaixo:

$$Y\_{k}=0.09512Y\_{k-1}+0.0488U\_{k} onde: Y\_{k-1}=0$$

Tabela 2 Obtenção dos valores do sistema por recursividade do sistema discreto

|  |  |  |  |
| --- | --- | --- | --- |
| K | U(k) | Y(k) | Y(k) |
| 0 | 0 | 0,9512\*0 + 0.0488\*0 | 0 |
| 1 | 1 | 0,9512\*0 + 0,0488\*1 | 0.0488 |
| 2 | 1 | 0,9512\*0.0488 + 0,0488\*1 | 0,0952 |
| 3 | 1 | 0,9512\*0,0952 + 0,0488\*1 | 0,1440 |
| 4 | 1 | 0,9512\*0 + 0,0488\*1 | 0,1858 |
| 5 | 1 | 0,9512\*0 + 0,0488\*1 | 0,2255 |
| 6 | 1 | 0,9512\*0 + 0,0488\*1 | 0,2633 |

Essa mesma ideia foi aplicada na rede neural. Definindo como entrada os valores de U obtidos pela PRBS e o valor das Saídas anteriores, como o sistema é de ordem um, o Y[k-1] é suficiente para identificá-lo. O fato de considerarmos o sistema de primeira ordem significa que não é uma caixa preta e na verdade é uma caixa cinza, onde já há algum conhecimento prévio. Os valores de Y[-k]:{k>0}, não são conhecidos, nesse caso consideramos que o sistema parte do repouso, assim todos os valores de Y[k] para k<0 são nulos.

A fase de treinamento da RNA inicia-se aplicando o valor U[0] tanto a função simulada (conhecida), quanto à rede neural, juntamente com as saídas anteriores. Essas entradas provocam uma reação na saída de ambas. O erro é calculado pela diferença entre a saída da rede e a saída do sistema simulado. O valor do erro é utilizado para ajustar os ganhos da rede, a fim de ajustar sua saída a saída do sistema. O processo continua para o período determinado e repete para cada época de aprendizado da rede.

Normalmente as RNAs são configuradas para aprender qual entrada gera uma saída. No entanto, como vimos anteriormente, na entrada em degrau o valor fica constante durante a maior parte do período, mas mesmo assim a saída continua variando até atingir o regime permanente. Se fixarmos a entrada, como no sinal de degrau após a transição, e alterarmos a saída durante a fase de treinamento, a rede neural sobrescreveria os pesos encontrado para o par entrada-saída anteriores, ou seja, a rede apenas aprenderia o valor de regime permanente, de modo que a identificação seria idêntica ao sinal de entrada e diferente da função do sistema. Para solucionar esse problema das redes neurais são utilizadas as técnicas de recursividade das saídas, para garantir que a RNA aprenda o comportamento da função de transferência real e não qual entrada gera aquela saída. Na condição da entrada U constante, temos saída anterior (Y[k-1]) realimentada na entrada variando, o que provoca uma variação da saída Y[k], que no próximo período será utilizada para gerar a Y[k+1].

A identificação de sistemas com redes neurais não tem como objetivo determinar os parâmetros ou o modelo matemático do sistema real, o objetivo é obter o modelo neural do sistema. O modelo neural tem a capacidade de reproduzir todas as respostas da planta real, mas o conhecimento a respeito da planta está intrinsicamente nos ganhos da rede, o qual não tem nenhuma ligação com o modelo matemático, são apenas combinações de entradas para criar a resposta idêntica a real.

Para entender o que é identificar, vejamos os significados em dois dicionários o Aurélio e o Michaelis.

 “Tornar idêntico, provar ou reconhecer a identidade de, absorver em si” (Aurélio).

“Tornar ou declarar idêntico; considerar duas coisas como idênticas, dando a uma o caráter da outra.” (Michaelis).

O modelo neural torna-se idêntico ao sistema real, absorvendo as características do mesmo em si. Isso prova que a obtenção do modelo neural, mesmo não revelando explicitamente os paramentos da função de transferência, é considerada uma identificação de sistema.

O algoritmo genético parte da otimização dos valores dos parâmetros da função de transferência. Como o sistema é caixa preta, nenhum parâmetro é conhecido, portanto os valores são gerados aleatoriamente, dentro dos limites impostos, normalizados entre zero e um.

Os parâmetros aleatórios são evoluídos pelo algoritmo até que sejam obtidos os melhores valores. Os parâmetros identificados não precisam ser idênticos aos reais, um erro desprezível pode ser considerado nulo, e os parâmetros encontrados formam a função de transferência identificada.

O sistema simulado é comparado com o sistema genético, obtendo o erro das saídas. Por meio desses erros é calculado o erro médio quadrático para determinar distância entre o sistema real e genético.

A função de aptidão precisa determinar quão próximo da solução ideal os parâmetros estão. Essa deve retornar um para a melhor solução e zero para a pior, utilizado o erro médio quadrático como indicador de qualidade. Abaixo vemos a função escolhida.

 $Fitness=\frac{1}{1+EMQ} $ & $EMQ=\sqrt{\frac{\sum\_{i}^{n}\left(Y\_{i\_{Real}}-Y\_{i\_{iden}}\right)^{2}}{n}}$

Considerando o pior caso de erro, que EMQ tenha um valor muito alto, o valor retornado pela função de aptidão será aproximadamente o inverso desse valor, ou seja, um valor muito próximo à zero. A melhor condição é o erro nulo, nessa condição as saídas reais e identificadas coincidem e a função de aptidão aproxima de um.

$Fitness\left(0\right)=\frac{1}{1+0}=\frac{1}{1}=1$

$$Fitness\left(\infty \right)=\frac{1}{1+\infty }=\lim\_{n\to \infty }\left(\frac{1}{1+n}\right)=0$$

A função evolve() foi criada para aplicar a função principal do algoritmo genético. Na primeira execução dessa função não há valores das aptidões calculadas dos indivíduos da população, portando deve-se calcular o valor para cada indivíduo e assim prosseguir para a função de cruzamento (*crossover*).

{

 Função de evolução da população do AG

}

**procedure** TGeneticAlg**.**evolve**();**

**begin**

 newPop**:=**TGAPop**.**Create**;** // aloca espaço para a

 // nova população

 **if** ffitness**.**Count **=** 0 **then** // primeira evolução?

 evaluatePopFitness**();** // avalia a aptidão

 crossover**();** // seleciona e cruza os

 // indivíduos

 mutate**();** // muta os indivíduos

 Inc**(**fcurrGeneration**);** // incrementa contador

 // de gerações

 evaluatePopFitness**();** // avalia a aptidão

**end;**

Assim como na natureza, o cruzamento não ocorre em todas as seleções de indivíduos, ela segue um probabilidade de ocorrência, definida aqui como um valor real entre 0 e 1 (100%), denominada probabilidade de cruzamento (PC).

Para que haja o cruzamento é necessário que um valor aleatório, também limitado entre 0 e 1, seja menor que o valor da PC, caso contrário os pais substituirão os filhos na nova população.

O algoritmo de cruzamento realiza a troca de características dos parâmetros do sistema, baseado no cruzamento de um ponto. Os parâmetros são codificados em valores inteiros de 10bits, logo o índice de crossover poderia ser um valor aleatório entre zero e nove, mas não faz sentido usar índice igual à zero ou igual a nove, pois a troca resultaria, nos mesmos indivíduos cruzado, isso significa que o sorteio deve ser entre um e oito.

O próximo passo é a seleção de qual parâmetro passará por cruzamento, para isso, sorteia qual dos parâmetros irá cruzar. Sortear o parâmetro evita que os parâmetros próximos ao ideal sejam perdidos na tentativa de encontrar os outros, que ainda estão distantes.

**procedure** TGeneticAlg**.**crossover**();**

**...**

**begin**

 evaluateProbabilities**();** // calcula os pesos da roleta

 // de acordo com as probabilidades

 **while(**newPop**.**Count **<>** fpopsize**)do** // completa toda a população

 **begin**

 parent1Idx **:=** self**.**selectIndividual**();** // seleciona indivíduo pela

 parent2Idx **:=** self**.**selectIndividual**();** // roleta com pesos

 randomize**();** //escolhe aleatoriamente

 **if(**Random **<=** fpc**)then** // se hover cruzamento

 **begin**

 crossIdx**:=** rand**(**1**,**fbitLen**-**1**);** //seleciona o bit de cruzamento

 mask **:=** **(** 1 **shl** crossIdx**)-**1**;** // cria a máscara

 NotMask **:=** **((**1 **shl** fbitLen**)-**1**)** **and** **(not** mask**);**

 childi1 **:=** TGACromo**.**Create**;** // cria os objetos dos filhos

 childi2 **:=** TGACromo**.**Create**;**

// varre todos os alelos dos cromossomos selecionados

 **for** j **:=** 0 **to** fpopulation**[**parent1Idx**].**Count**-**1 **do**

 **begin**

 p1 **:=** fpopulation**[**parent1Idx**][**j**];**

 p2 **:=** fpopulation**[**parent2Idx**][**j**];**

 // cruza os dois para originar dois decendentes

 childi1**.**Add**((**p1 **and** mask**)** **or** **(**p2 **and** NotMask**));**

 childi2**.**Add**((**p1 **and** NotMask **)** **or** **(**p2 **and** mask**));**

 **end;**

 **end**

 **else**

 **begin**

 // caso não haja cruzamento

 // os pais ocuparão o lugar dos filhos

 childi1**:=** fpopulation**[**parent1Idx**];**

 childi2**:=** fpopulation**[**parent2Idx**];**

 **end;**

 newPop**.**Add**(**childi1**);** // os filhos são adicionados

 newPop**.**Add**(**childi2**);** // à nova população

 **end;**

 fpopulation**:=** newPop**;**

**end;**

Os novos indivíduos gerados pelo cruzamento e os pais que não cruzaram se unem para formar a nova população. Então o critério de parada é avaliado, ou seja, verifica-se se a aptidão do melhor indivíduo está acima do valor aceitável. Em caso afirmativo o algoritmo se encerra; caso contrário uma nova geração é criada.

# Analise dos métodos estudados

## Sistema

O sistema é representado por uma função de transferência de segunda ordem, no domínio discreto, por meio de amostragem. O período de amostragem de 50ms foi calculado para que o sistema não seja subamostrado e nem sobreamostrado, pois isso poderia influenciar a comparação dos métodos. O sistema abaixo foi escolhido para compará-los.

$$Y\left[k\right]= 1,4138∙Y\left[k-1\right]- 0,6065∙Y\left[k-2\right]$$

$$+0,000∙U\left[k-1\right]+0.0626∙U\left(k-2\right)+0.0530∙U\left(k-3\right)$$

Para que os métodos não tenham acesso aos paramentos da função, criamos um vetor contentando apenas as entras e as saídas geradas correspondentes às entradas. Abaixo vemos a função responsável por gerar esses valores.

// Gera o valor de Y para o sistema no modelo abaixo

//

// Y[k]:= a0.U[k-1] + a1.U[k-2]+ a2.U[k-3] - b0.Y[k-1] - b1.Y[k-2]

//

**procedure** CreateSystem**(**a0**,**a1**,**a2**,**b0**,**b1**:**Extended**;var** IO**:**TIO**);**

**var**

 i**,**sz**:** Integer**;**

 // função que retorna o valor da saída em "k"

 **function** Y**(**k**:**integer**):**Extended**;**

 **begin**

 **if** k**<**0 **then**

 Result**:=** 0

 **else**

 Result**:=** IO**[**k**].**Y**;**

 **end;**

**begin**

 sz**:=** length**(**IO**);** // número máximo de amostragens

 **for** i **:=** 0 **to** sz**-**1 **do**

 **begin**

 // cálculo do Y atual

 IO**[**i**].**Y **:=** a0**\***U**(**i**-**1**)** **+**a1**\***U**(**i**-**2**)** **+**a2**\***U**(**i**-**3**)** **+**b0**\***Y**(**i**-**1**)** **-**b1**\***Y**(**i**-**2**);**

 // a função U é definida como a entrada

 // PRBS para treinamento

 // Degrau para execução

 **end;**

**end;**

Durante a fase de treinamento, o vetor de entrada foi preenchido com o sinal gerado pela função PRBS. Esse sinal induz um grande espectro de frequências no sistema facilitando o aprendizado das características dos sistemas físicos.

Figura 16 Sinal de entrada PRBS, usado durante o treinamento.

Fonte: acervo do autor.

Na fase de validação, utilizamos a função degrau discreta. Essa função é amplamente usada para induzir distúrbios em controles. Sua definição é apresentada abaixo:

$$U\left(k\right)=0, se k\leq 0$$

$$U\left(k\right)=1, se k>0$$

É importante lembrar que o valor de “k” é sempre um número inteiro, já que estamos trabalhando com sistemas discretos.

Figura 17 Sinal de entrada Degrau unitário, usado na validação dos métodos.

As redes neurais possuem dois parâmetros principais que influenciam na convergência do ajuste dos ganhos durante a fase de aprendizado. Esses parâmetros são o coeficiente de aprendizado, $α$, e o coeficiente de momento, $μ$.

Figura 18 Gráfico representando a relação entre o fator de aprendizado, momento e o EMQ.

Fonte: acervo do autor.

Para determinar o melhor valor de cada um dos coeficientes da rede neural, uma curva de superfície foi gerada para analisar a influência destes no erro da rede. Na superfície pode-se notar uma região plana, com $α<0,73$. Nessa região o valor do erro permanece aproximadamente constante, o que significa que o momento quase não influencia na convergência do erro, para altas taxas de aprendizado. A exceção ocorre para valores de momento muito baixos, pois nesses casos o momento ajuda a aumentar velocidade de convergência.

Seguindo pela superfície notamos um vale, onde os valores de erro são minimizados, mais precisamente para $0,8\leq α\leq 0,9$. Acima de 0,9 a rede torna-se instável, oscilando entre os mínimos locais do erro médio quadrático e não convergindo para o mínimo global. Esse efeito reduz quando o momento é baixo.

Após a determinação das regiões ótimas, as relações entre cada coeficiente e a análise de vários testes, os valores que minimizavam o erro e ainda mantinham o sistema estável foram $μ=0,3$ e $α=0,8$.

Finalizando a calibração de cada algoritmo para obter o máximo aproveitamento, iniciamos a comparação dos métodos para identificação de sistemas.

Abaixo temos os resultados para a rede neural. Esse método consegue aproximar do sistema real, no entanto apresenta alguns desvios que são reduzidos quando há um aumento do no numero de épocas de treinamento.

Figura 19 Identificação do sistema para 50 épocas de treinamento.

Fonte: acervo do autor.

Figura 20 Identificação do sistema para 100 épocas de treinamento.

Fonte: acervo do autor.

Figura 21 Identificação do sistema para 250 épocas de treinamento

Fonte: acervo do autor.

Figura 22 Sistema real (vermelho) e sistema identificado pela AG (azul).

Fonte: acervo do autor.

Os resultados deixam claro que a eficiência do sistema melhora, ou seja, reduz o erro, com o aumento do número de treinamentos. O gráfico abaixo mostra essa relação.

Figura 23 Erro em função do número de épocas de treinamento.

Fonte: acervo do autor.

Em algoritmos genéticos o que mais se assemelha às épocas de treinamento da RNA é a geração. A geração representa uma iteração do algoritmo para gerar uma nova população de soluções. A cada geração do Algoritmo Genético as soluções vão evoluindo, eliminando-se as soluções ruins e combinando as melhores, tornando a população cada vez mais próxima do conjunto das soluções ideais.

Figura 24 Identificação por AG, geração 50.

Fonte: acervo do autor.

Figura 25 Identificação por AG, geração 100.

Figura 26 Identificação por AG, geração 250.

Fonte: acervo do autor.

Figura 27 Identificação por AG, geração 500.

Fonte: acervo do autor.

Abaixo vemos um resumo dos resultados e a comparação dos dois métodos.

Tabela 3 - Comparação dos métodos.

|  |  |  |
| --- | --- | --- |
| **Característica** | **Algoritmo Genético** | **Rede Neural Artificial** |
| Erro Médio Quadrático (\*) | 13,759E-3 | 29,084E-3 |
| Linhas de Código(\*\*) | 581 | 1180 |
| Tipo de Identificação | Caixa-branca | Caixa-cinza |
| Limitação de entradas | Não | Não |
| Necessita de modelo | Sim | Não |
| Sistemas não lineares (\*\*\*)  | Sim | Sim |
| Treinamento | Não | Sim |
| Tempo de execução (\*) | 1313ms | 278ms |

# Conclusão

Conforme os resultados obtidos, ambos os algoritmos conseguem identificar o sistema de forma satisfatória. No entanto, usando o erro médio quadrático como referência, nota-se que o algoritmo genético obteve uma leve vantagem sobre a rede neural.

Usando como parâmetro o número de iterações a rede neural obteve vantagem, validando a retropropagação, que provê uma maior rapidez no ajuste dos pesos, ao passo que os métodos evolutivos gastam um tempo maior para obter ajuste equivalente.

Quanto à velocidade de execução, o algoritmo genético é o mais rápido, provavelmente devido ao fato desse não necessitar de derivadas para achar a solução ótima.

A rede neural possui a limitação de não encontrar os parâmetros do sistema. Apesar disso, notamos que ela consegue identificar o sistema de forma eficiente. Talvez essa limitação não seja uma restrição em trabalhos de pesquisadores que queiram utilizar esse algoritmo.

O Algoritmo Genético tem sua eficiência limitada pela função de aptidão. Uma função de aptidão baseada apenas no erro médio quadrático não foi suficiente para encontrar os valores exatos da função, mas a inclusão da derivada na avaliação da aptidão melhora significativamente o processo de identificação.

Quanto à estabilidade, o algoritmo genético provou ser mais instável, o que era esperado, pois o algoritmo genético baseia suas soluções em probabilidades, podendo encontrar ótimos locais e, consequentemente, gastando um número maior de iterações para retornar a solução ideal. A rede neural baseia no treinamento, quanto maior for o número de épocas no treinamento maior será a estabilidade do sistema. Esse é o motivo da ampla utilização de redes neurais para identificar sistemas não lineares.

Portanto, a escolha do melhor método para aplicar na identificação de sistemas vai depender das restrições impostas em projeto, como numero de iterações, erro máximo aceitável, complexidade do método para implementação em sistemas embarcados, por exemplo, e o mais importante: o tipo de modelo esperado, ou seja, a identificação por determinação de parâmetros ou o modelo neural do sistema.

Trabalhos futuros poderão estudar a combinação dos métodos empíricos de identificação de sistemas para a formação de métodos híbridos para identificar sistemas não lineares, validando a melhoria alcançada e verificando se esses métodos poderiam substituir os métodos de identificação matemáticos.

Outra implantação futura seria a implementação de um controle inteligente, que possua a capacidade de identificar o sistema controlado, e a partir dessa informação, determina qual é o melhor controle para esse, tornando-o mais robusto e imune a variações da função de transferência da planta provocado por fatores físicos, como a troca de componentes, acumulo de resíduos, desgaste, matéria prima entre outros.

REFERÊNCIAS BIBLIOGRÁFICAS

CERQUEIRA, Eduardo O. de Andrade; et al. **Redes neurais e suas aplicações em calibração multivariada**. 2001. Química Nova – Universidade de Franca, São Paulo.

ROCHA, F. Lopes. **Identificação de sistemas não lineares multivariáveis usando redes neurais Perceptron Multicamadas e Função de Base Radial**. 2006. Dissertação (mestrado em Engenharia da Produção e Sistemas) – Pontifícia Universidade Católica do Paraná, Paraná.

BARRETO, Jorge M. **Introdução às Redes Neurais Artificiais.** 2002. Laboratório de Conexionismo e Ciências Cognitivas, UFSC – Universidade Federal de Santa Catarina, Florianópolis.

JOVANOVIC, Olivera. **Identification of dynamic system using neural network**. 1998. The Scientific Journal Facta Universitatis – University of Niš, Montenegro.

COELHO, Leandro dos Santos; MARIANI, Viviana Cocco. **Sistema híbrido neuro-evolutivo aplicado ao controle de um processo multivariável**.2006. Sba: Controle & Automação Sociedade Brasileira de Automática - Universidade Católica do Paraná.

Zibo, Z.; Naghdy, F. **Application of Genetic Algorithms for system identification.** 1995. IEEE – International Conference on Evolutionary Computing, Vol. 2. University of Wollongong – Australia.

TANOMARU, Julio. **Motivação, fundamentos e aplicações de algoritmos genéticos.** Anais do II Congresso Brasileiro de Redes Neurais. Vol. 1. 1995.

Ssu-Hsin Yu. **Model-based Identification and Control of Nonlinear Dynamic Systems Using Neural Network.** 1996. Dissertação (Doutorado em Engenharia Mecânica) – Massachusetts Institute of Technology, EUA.

VELLASCO, Marley Maria Bernardes Rebuzzi. **Redes Neurais Artificiais.** 2007. Laboratório de Inteligência Computacional Aplicada – Pontifica Universidade Católica do Rio de Janeiro.

Kumon, Toshiro; Iwasaki, Makoto; Suzuki, Tatsuya; Hashiyama, Tomonori;

Matsuit, Nobuyuki; Okumat, Shiger. Nonlinear Systenl **Identification Using Genetic Algorithm**. 2000. IEEE – Nagoya Institute of Technology, Nagoya, Japão.

Leão, Marcelo. **Borland Delphi 7 Curso Completo**. Rio de Janeiro: Axcel Book do Brasil Editor Ltda. 2003. 807p.

MEDEIROS, L. F. de. **Redes Neurais em Delphi**. Florianópolis: Visual Books, 2006. 210p.

SIMON S. HAYKIN. **Redes Neurais - 2ed.** Bookman Companhia Ed, 900p.