



ÁLVARO HENRIQUE NOGUEIRA DE LIMA

**SELEÇÃO DE CARACTERÍSTICAS DE
DADOS UTILIZANDO REDES NEURAS
ARTIFICIAIS**

**LAVRAS – MG
2012
ÁLVARO HENRIQUE NOGUEIRA DE LIMA**

**SELEÇÃO DE CARACTERÍSTICAS DE DADOS UTILIZANDO
REDES NEURAIS ARTIFICIAIS**

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Sistemas de Informação para obtenção do título de Bacharel em Sistemas de Informação.

Orientador

Dr. Wilian Soares Lacerda

LAVRAS – MG

2012

ÁLVARO HENRIQUE NOGUEIRA DE LIMA

**SELEÇÃO DE CARACTERÍSTICAS DE DADOS
UTILIZANDO REDES NEURAIS ARTIFICIAIS**

Monografia de graduação apresentada ao
Colegiado do Curso de Sistemas de
Informação, para obtenção do título de
Bacharel em Sistemas de Informação.

APROVADA em 10 de outubro de 2012.

ANDRÉ GRÜTZMANN

DANTON DIEGO FERREIRA


WILIAN SOARES LACERDA (orientador/a)

(Co-Orientador)

LAVRAS-MG

2012

AGRADECIMENTOS

À Universidade Federal de Lavras e ao Departamento de Ciência da Computação por fornecer a estrutura e recursos necessários para a realização da Graduação.

Ao professor Dr. Wilian Soares Lacerda pela orientação, paciência e ensinamentos transmitidos que foram de grande importância para a realização deste trabalho.

Ao meu colega Alan pela preciosa ajuda no decorrer de toda a graduação.

RESUMO

As Redes Neurais Artificiais tem se difundido ao longo dos anos e sua utilização vem crescendo devido aos bons resultados encontrados na solução de diversos problemas do mundo real. Porém o elevado grau de complexidade presente nas Redes Neurais Artificiais torna sua computação difícil, um dos fatores que prejudicam o desempenho da rede neural é a presença de variáveis de entrada redundantes que nada acrescentam ao seu processo de aprendizagem tornando assim o treinamento mais difícil e demorado. Os métodos de seleção de características têm por objetivo determinar quais variáveis (características) da entrada são mais relevantes para a determinação da saída ou resposta da rede, isto possibilita a redução do número de entradas da rede. Neste trabalho foram implementados cinco métodos de seleção de características, o método de Garson, Perturb, PaD, Análise de Sensibilidade e Correlação. Foram escolhidos três problemas (Iris, CPU Performance, Resistência do concreto) para o treinamento das redes neurais, após treinadas com o algoritmo backpropagation os métodos foram executados obtendo-se a importância de cada entrada, as entradas menos importantes foram excluídas e as redes retreinadas obtendo-se um novo erro médio quadrático que foi comparado ao original de forma a avaliar o desempenho do método. Para o problema da Iris considerado mais simples todos os métodos obtiveram resultados semelhantes. Já para os problemas mais complexos com a presença de mais variáveis como o da CPU Performance e Resistência do Concreto os métodos Perturb e Correlação apresentaram os piores resultados, o método de Garson obteve um resultado satisfatório e o PaD e Análise de Sensibilidade apresentaram melhores resultados e se destacaram em relação aos demais.

Palavras-chave: Redes Neurais Artificiais. Treinamento. Backpropagation. Seleção de Características. Importância. Variáveis.

ABSTRACT

Artificial Neural Networks have been disseminated over the years and its use has grown because of the good results in solving many real world problems. But the high degree of complexity present in Artificial Neural Networks makes its computation difficult, one of the factors that affect the performance of the neural network is the presence of redundant input variables that add nothing to their learning process thus making training more difficult and time consuming. The feature selection methods aim to determine which variables (characteristics) of the input that are most relevant or most contribute to the determination of output or response of the network, this enables the reduction of the number of entries to be excluded from the network variables less significant. In this work we implemented five methods of feature selection, the method of Garson, perturb, DBP, Sensitivity Analysis and Correlation. They chose three problems (Iris, CPU Performance, Strength of Concrete) for training the networks trained with the backpropagation algorithm following methods were performed yielding the importance of each entry, the entries were excluded and less important networks retrained obtaining A new mean square error which was compared to the original in order to evaluate the performance of the method. For the problem considered simpler Iris all methods yielded similar results. But for the most complex problems with the presence of more variables such as CPU Performance and Strength of Concrete methods perturb and Correlation showed the worst results, the method of Garson obtained a satisfactory result and Pad and Sensitivity Analysis showed better results and highlighted in relation to others.

Keywords: Artificial Neural Networks. Training. Backproagation. Feature Selection. Importance. Variables.

LISTA DE FIGURAS

Figura 1 Estrutura de um neurônio biológico	16
Figura 2 Modelo de neurônio proposto por McCulloch e Pitts.....	17
Figura 3 Exemplos funções de ativação	18
Figura 4 Rede feedforward de uma única.....	20
Figura 5 Rede feedforward de múltiplas camadas	21
Figura 6 Exemplo de uma rede neural recorrente	22
Figura 7 Exemplo de uma rede MLP	23
Figura 8 Mecanismo de aprendizado supervisionado	24
Figura 9 Mecanismo de aprendizado não supervisionado.....	26
Figura 10 Fase <i>forward</i> do algoritmo backpropagation	29
Figura 11 Fase <i>backward</i> do algoritmo backpropagation	29
Figura 12 Tela inicial do Scilab.....	40

LISTA DE GRÁFICOS

Gráfico 1 Erro médio quadrático para os dados de treinamento e teste em função do número de épocas.....	47
Gráfico 2 Importância das variáveis de acordo com o algoritmo de Garson	48
Gráfico 3 Correlação entre as variáveis do problema Iris	49
Gráfico 4 Erro médio quadrático em função da adição de ruído nas variáveis de entrada	49
Gráfico 5 Contribuição Relativa de cada variável de acordo com o método PaD.....	50
Gráfico 6 Contribuição de cada variável de acordo com o método de Análise de Sensibilidade	50
Gráfico 7 Erro médio quadrático para os dados de treinamento e teste em função do número de épocas.....	52
Gráfico 8 Importância das variáveis de acordo com o algoritmo de Garson	53
Gráfico 9 Correlação entre as variáveis.....	54
Gráfico 10 Erro médio quadrático em função da adição de ruído nas variáveis de entrada.....	54
Gráfico 11 Contribuição de cada variável de acordo com o método de Análise de Sensibilidade	55
Gráfico 12 Contribuição Relativa de cada variável de acordo com o método de PaD	55
Gráfico 13 Erro médio quadrático para os dados de treinamento função do número de épocas.....	57
Gráfico 14 Importância das variáveis de acordo com o algoritmo de Garson	59
Gráfico 15 Correlação entre as variáveis.....	59
Gráfico 16 Erro médio quadrático em função da adição de ruído nas variáveis de entrada.....	60
Gráfico 17 Contribuição Relativa de cada variável de acordo com o método de PaD	60

Gráfico 18 Contribuição de cada variável de acordo com o método de Análise de Sensibilidade	61
---	----

LISTA DE TABELAS

Tabela 1 - Importância das variáveis de acordo com cada método de seleção de características para o problema da Iris.....	47
Tabela 2 - Importância das variáveis de acordo com cada método de seleção de características para o problema CPU Performance.	52
Tabela 3 - Resultado do retreinamento da RNA – CPU Performance.	56
Tabela 4 - Importância das variáveis de acordo com cada método de seleção de características para o problema da Resistência do Concreto.....	58
Tabela 5 - Resultado do retreinamento da RNA – Resistência Concreto....	62

SUMÁRIO

1 INTRODUÇÃO	11
1.1 Motivação	11
1.2 Objetivos.....	12
1.3 Estrutura do Trabalho	12
2 REDES NEURAIS ARTIFICIAIS.....	14
2.1 Introdução	14
2.2 Breve Histórico	14
2.3 Neurônio Biológico	15
2.4 Neurônio Artificial	16
2.5 Arquiteturas de Rede	19
2.5.1 Redes Feedforward de camada única	19
2.5.2 Redes Feedforward de múltiplas camadas	20
2.5.3 Redes Recorrentes	21
2.6 Rede MultiLayer Perceptron (MLP)	22
2.7 Aprendizado das Redes Neurais.....	23
2.7.1 Aprendizado supervisionado.....	24
2.7.2 Aprendizado não supervisionado	25
2.7.3 Aprendizado por reforço	26
2.8 Regra Delta	27
2.9 Algoritmo Error Backpropagation.....	28
2.10 Dificuldades do treinamento	32
3 SELEÇÃO DE CARACTERÍSTICAS	33
3.1 Algoritmo de Garson	33
3.2 Análise de Sensibilidade.....	34
3.3 Método <i>PaD</i>	36
3.4 Método Perturb	37
3.5 Correlação.....	37
4 METODOLOGIA	39
4.1 Ambiente de Programação (Scilab).....	39

4.2 Treinamento das Redes Neurais	40
4.2.1 Pré-processamento dos dados	40
4.2.2 Configuração da rede.....	41
4.3 Base de dados.....	42
4.3.1 Performance Relativa da CPU	43
4.3.2 Iris.....	43
4.3.3 Resistência do concreto	44
4.4 Análise dos resultados	45
5 RESULTADOS.....	46
5.1 Iris	46
5.1.1 Definição da importância das variáveis de entrada	46
5.1.2 Retreinamento da RNA.....	50
5.2 CPU Performance.....	51
5.2.1 Definição da importância das variáveis de entrada	51
5.2.2 Retreinamento da RNA.....	55
5.3 Resistência do Concreto	56
5.3.1 Definição da importância das variáveis de entrada	57
5.3.2 Retreinamento da RNA.....	61
6. CONCLUSÃO.....	63
REFERÊNCIAS.....	65
ANEXO A – Algoritmo de Garson.....	67
ANEXO B – Método PaD	68
ANEXO C – Análise de Sensibilidade.....	69

1 INTRODUÇÃO

Redes neurais artificiais fazem parte de uma classe da inteligência artificial cujo funcionamento tenta reproduzir uma rede de neurônios natural (cérebro) de forma simplificada.

Ao longo dos anos sua utilização vem sendo difundida devido aos bons resultados encontrados na solução de diversos problemas do mundo real e também devido à evolução do hardware e software que tornaram sua implementação mais intuitiva e rápida.

Diversos são os tipos de problemas nos quais as redes neurais podem ser aplicadas, dentre eles podem-se destacar principalmente os problemas de classificação, clusterização, reconhecimento de padrões e aproximação de funções.

1.1 Motivação

O elevado grau de complexidade presente nas redes neurais artificiais torna sua computação difícil, e em muitos casos, seu tempo de treinamento e custo computacional se torna muito elevado. Um dos fatores que prejudicam o desempenho da rede neural é a presença de variáveis de entrada redundantes que nada acrescentam ao seu processo de aprendizagem tornando assim o seu treinamento mais difícil e demorado.

Visando contornar esses problemas muitas técnicas e métodos de seleção características de dados utilizando Redes Neurais Artificiais vêm sendo propostos ao longo do tempo. Estes métodos têm por objetivo determinar quais variáveis (características) da entrada são mais relevantes ou que mais contribuem para a determinação da saída ou resposta da rede neural.

Através da seleção de características é possível eliminar as variáveis menos importantes e reduzir a dimensão dos dados de entrada fazendo com que o treinamento da rede neural seja mais rápido e eficiente.

1.2 Objetivos

O objetivo geral deste trabalho é fazer um levantamento de algumas técnicas de seleção de características utilizando Redes Neurais Artificiais encontradas na literatura, estudá-las e implementá-las.

Também é objetivo deste trabalho fazer uma análise comparativa entre estes métodos, de forma a determinar qual método apresenta melhores resultados na determinação da importância das entradas da rede para diversos tipos de problemas.

1.3 Estrutura do Trabalho

O Capítulo 2 apresenta uma breve revisão de literatura sobre redes neurais artificiais, com ênfase na rede MLP (Multi Layer Perceptron) e seu algoritmo de treinamento *backpropagation*, os quais serão utilizados em todas as redes neurais implementadas neste trabalho.

O Capítulo 3 traz uma revisão bibliográfica sobre alguns métodos de seleção características de dados utilizando Redes Neurais Artificiais encontrados na literatura, entre eles estão o método de Garson, baseado no produto dos pesos das conexões. O método Pertub avalia o efeito de pequenas mudanças em cada variável de entrada na saída da rede neural. E por fim os métodos de análise de sensibilidade, o PaD baseado na derivada parcial da saída da rede em relação a entrada, e o método Valença e Ludemir que tem como base as derivadas parciais da variável de saída em relação aos pesos das conexões.

No Capítulo 4 serão apresentados os materiais e métodos utilizados na realização deste trabalho, incluindo uma breve descrição dos problemas escolhidos e do ambiente de programação utilizado (Scilab) juntamente com o seu *toolbox* de redes neurais. Será explicado também como foi feita a preparação dos dados, configuração e treinamento das redes neurais utilizadas neste trabalho.

O Capítulo 5 apresenta os resultados obtidos dos métodos apresentado neste trabalho para cada problema. E por fim o capítulo 6 apresenta as conclusões finais acerca deste trabalho.

2 REDES NEURAIIS ARTIFICIAIS

Este capítulo apresenta os conceitos básicos de redes neurais artificiais necessários para o desenvolvimento deste trabalho, entre eles destacam-se as redes MLP (*MultiLayer Perceptron*) e seu algoritmo de treinamento o *backpropagation*.

2.1 Introdução

Redes Neurais Artificiais (RNAs) podem ser consideradas como uma técnica utilizada em resolução de diversos tipos de problemas característicos da Inteligência Computacional (IC), dentre eles, previsão, classificação, reconhecimento de padrões e clusterização. As RNAs são inspiradas no funcionamento do cérebro humano procurando-se, segundo Barreto (1997), construir um computador que tenha circuitos modelando os circuitos cerebrais esperando-se ver um comportamento inteligente emergindo, aprendendo novas tarefas, errando, fazendo generalizações e descobertas, e frequentemente ultrapassando seu professor.

Segundo Haikin (1999, p.28) uma rede neural artificial pode ser definida como: “um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental”.

2.2 Breve Histórico

O primeiro trabalho a propor um modelo de computação baseado nos neurônios biológicos foi realizado pelos pesquisadores McCulloch e Pitts (1943), na ocasião sugeriram um modelo de neurônios artificiais em que cada neurônio se caracteriza por estar “ligado” ou “desligado”, e este estado mudaria para “ligado” em decorrência à estimulação por um número suficiente de neurônios vizinhos.

Outros importantes trabalhos foram apresentados ao longo dos anos. Hebb (1949) propôs a primeira regra para aprendizagem auto-organizada assumindo que a aprendizagem do conhecimento representado em uma rede neural seja atingida pelo fortalecimento das conexões entre neurônios adjacentes, sempre que esses estiverem excitados. Rosenblatt (1958) propôs o perceptron, sendo este considerado como o primeiro modelo para aprendizado supervisionado.

2.3 Neurônio Biológico

O neurônio biológico é constituído principalmente por:

1. Os dendritos, os quais tem por função receber os estímulos transmitidos pelos outros neurônios;
2. O corpo de neurônio, também chamado de soma, que é responsável por coletar e combinar informações vindas de outros neurônios;
3. E finalmente o axônio, que é constituído de uma fibra tubular que pode alcançar até alguns metros, e é responsável por transmitir os estímulos para outras células.

Segundo Kovacs (2002) o neurônio biológico Figura 1 pode ser visto como o dispositivo computacional elementar do sistema nervoso, composto de várias entradas e uma saída. As entradas são formadas através das conexões sinápticas que conectam os dendritos aos axônios de outras células nervosas. Os sinais que chegam por estes axônios são pulsos elétricos conhecidos como impulsos nervosos ou potenciais de ação e constituem a informação que o neurônio processa para produzir como saída um impulso nervoso no seu axônio.

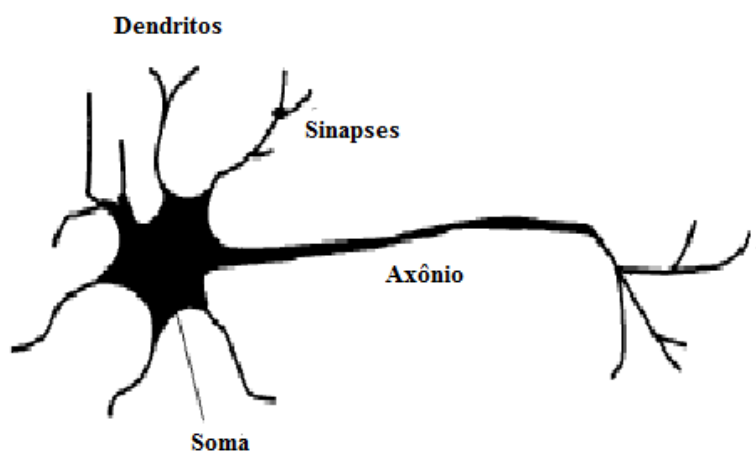


Figura 1 Estrutura de um neurônio biológico
Fonte: Veelenurf (1995)

2.4 Neurônio Artificial

Como citado anteriormente o primeiro modelo de um neurônio artificial foi proposto por McCulloch e Pitts (1943), este neurônio era uma representação muito simples do neurônio biológico e que de acordo com Kovacs (2002) era basicamente um dispositivo binário composto por uma saída (“pulso” ou “não pulso”) e várias entradas que poderiam ser excitatórias ou inibitórias. Para estabelecer a saída é realizada uma soma ponderada das entradas com o respectivo peso (positivos no caso excitatório e negativos nos casos inibitórios). Se o resultado desta soma fosse maior ou igual a certo limiar (threshold) a saída era “pulso” caso contrário “não pulso”.

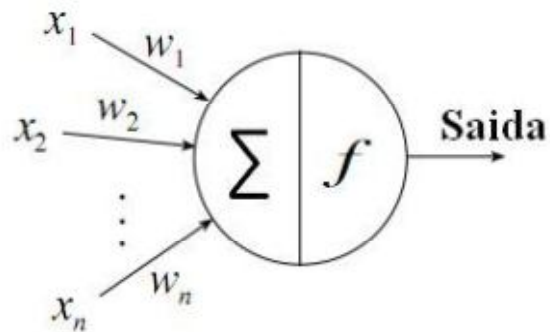


Figura 2 Modelo de neurônio proposto por McCulloch e Pitts

Na Figura 2 acima x representa as entradas do neurônio, w os pesos associados à conexão, Σ é a função que realizará a soma ponderada das entradas e f representa a função de transferência que tem o mesmo papel do limiar de disparo do neurônio biológico.

Finalmente, pode-se levantar algumas limitações no modelo original proposto por McCulloch e Pitts tais como:

1. Redes de neurônios de McCulloch e Pitts com apenas uma camada só conseguem implementar funções linearmente separáveis;
2. Pesos negativos são mais adequados para representar disparos inibidores;
3. O modelo foi proposto com pesos fixos, não-ajustáveis.

Com base no modelo de neurônio de McCulloch e Pitts diversos outros modelos foram derivados que possibilitam uma saída qualquer e com diferentes funções de ativação como, por exemplo: a função linear, Figura 3.a, a função rampa, Figura 3.b, a função degrau (*step*), Figura 3.c e a função sigmoideal, Figura 3.d.

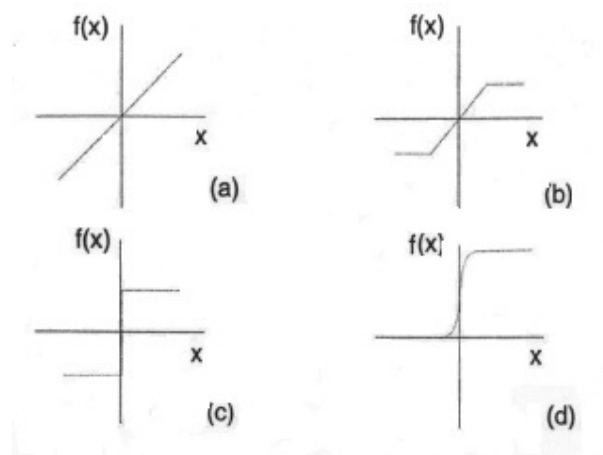


Figura 3 Exemplos funções de ativação

Em 1958 Franklin Rosenblatt propôs um dos modelos mais simples de Rede Neural, o *Perceptron*, sendo este modelo o primeiro a incluir o conceito de aprendizado. Apesar de possuir capacidade de aprendizado o *Perceptron* foi alvo de muitas críticas devido ao fato de possuir saídas com valores discretos, tendo dificuldade em trabalhar com valores que possuam ruídos e também não possuir capacidade de classificar dados não linearmente separáveis, Misk e Papert (1969) argumentaram ainda as limitações de perceptrons isolados como a incapacidade de resolverem alguns problemas como o XOR(ou exclusivo) por exemplo.

Simplificadamente o algoritmo de aprendizado do *Perceptron* consiste nos seguintes passos:

1. Inicializar pesos e threshold (limiar de ativação) com zero ou um valor aleatório próximo de zero.
2. Apresentar nova entrada que vai se somar a saída desejada.
3. Calcular saída atual.
4. Atualizar o peso. Esta atualização é feita com base na regra de correção dos erros mostrado na Equação (1):

$$w_2 = w_1 + \Delta w_1 = w_1 + \eta(d_1 - o_1)x_1 \quad (1)$$

Onde w_1 representa o vetor de pesos atuais, x_1 o vetor de entrada, d_1 a saída desejada, o_1 a saída calculada, a constante η a taxa de aprendizado e w_2 o novo vetor de pesos. O erro utilizado na equação é calculado através da diferença entre a saída desejada d_1 e a saída calculada o_1 .

Widrow e Hoff (1960) apresentaram um novo modelo de RNA chamada de ADALINE (ADAPtative LINEar Element), que permitia a utilização de uma função contínua na saída da rede.

O processamento do sinal na rede ADALINE ocorre da mesma forma que na rede *Perceptron*, porém a diferença está no cálculo da saída, pois é possível utilizar várias funções de ativação, das quais as mais utilizadas são a função linear, função Sigmoidal, função Logística e a função Tangente Hiperbólica.

2.5 Arquiteturas de Rede

A maneira como os neurônios de uma rede neural estão organizados e conectados uns aos outros neurônios é o que define a arquitetura da Rede Neural. Em geral, as RNAs estão enquadradas em três diferentes tipos de arquiteturas: feedforward de camada única, feedforward múltiplas camadas e redes recorrentes.

2.5.1 Redes Feedforward de camada única

Esta é a forma mais simples de se implementar uma rede em camadas, na qual a rede possui uma camada de entrada de nós que se conectam aos neurônios (nós computacionais) da camada de saída, mas não ao contrário, os sinais são transmitidos somente na direção da entrada para a saída, daí o nome feedforward (“alimentadas adiante”) (HAIKIN,1999).

Apesar de possuir “duas” camadas este tipo de rede neural é denominada de camada única devido ao fato da mesma possuir apenas uma camada de nós computacionais, desconsiderando-se assim a camada de entrada que somente propaga o sinal da entrada não realizando nenhum tipo de computação. A Figura 4 mostra um exemplo de rede feedforward de camada única.

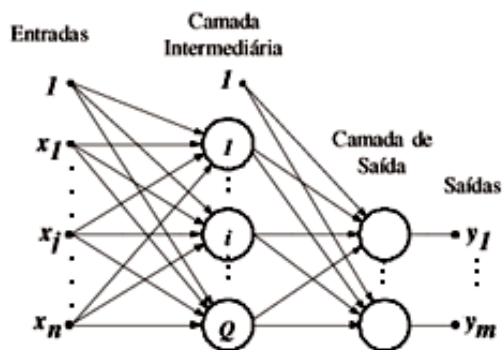


Figura 4 Rede feedforward de uma única
Fonte: Caloba (et., al 2002)

2.5.2 Redes Feedforward de múltiplas camadas

Rede feedforward de múltiplas camadas, Figura 5, como o próprio nome sugere possuem além da camada de entrada e saída (nós computacionais) pelo menos uma camada intermediária ou escondida em sua arquitetura, sendo que a camada de entrada propaga seu sinal para a camada escondida e a saída da camada escondida é propagada para a camada de saída.

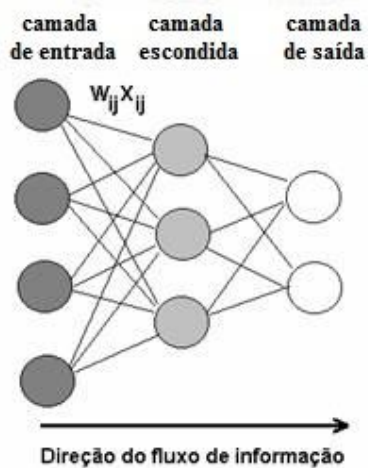


Figura 5 Rede feedforward de múltiplas camadas

As redes feedforward de múltiplas camadas ainda podem ser classificadas como *totalmente conectadas*, quando todos os nós ou neurônios de uma camada da rede estão conectados a todos os outros nós da camada adjacente. Porém quando há algum elo ou conexão inexistente entre um ou mais nós de uma camada com os nós da camada adjacente a rede é classificada como *parcialmente conectada* (HAIKIN, 1999).

2.5.3 Redes Recorrentes

As redes neurais recorrentes, Figura 6, se diferem das redes feedforward justamente pelo fato de possuírem pelo menos um laço de retroalimentação, ou seja, as entradas dos neurônios são realimentadas pelas saídas de todos os outros neurônios.

Segundo Haikin (1999) a presença de laços de retroalimentação nas redes recorrentes tem um impacto profundo na capacidade de aprendizagem e desempenho da rede.

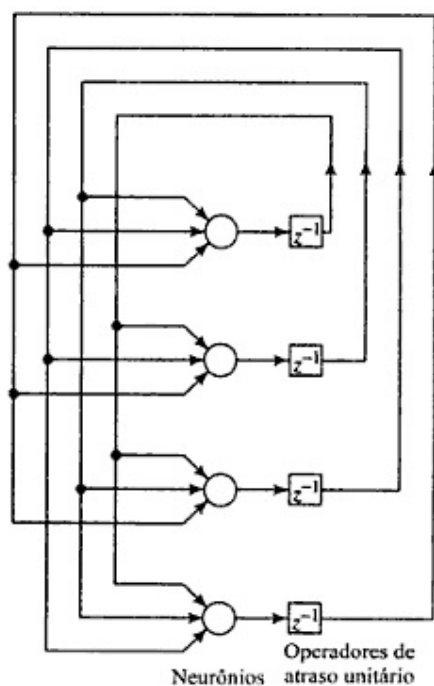


Figura 6 Exemplo de uma rede neural recorrente

Fonte: Haikin (1999)

2.6 Rede MultiLayer Perceptron (MLP)

As redes de uma única camada, como citado anteriormente, são capazes somente de resolverem problemas linearmente separáveis, as redes MLP consistem basicamente em uma camada de entrada, uma ou duas camadas intermediárias ou escondidas, e uma camada de saída e são capazes de implementar qualquer função contínua como demonstrado por Cybenko (1989).

Redes MLP são computacionalmente mais poderosas do que redes sem camadas intermediárias. Ao contrário destas, MLPs podem lidar com dados que não são linearmente separáveis. Teoricamente, redes com duas camadas intermediárias podem implementar qualquer função, seja ela

linearmente separável ou não (CYBENKO, 1989). A Figura 7 apresenta uma rede MLP típica.

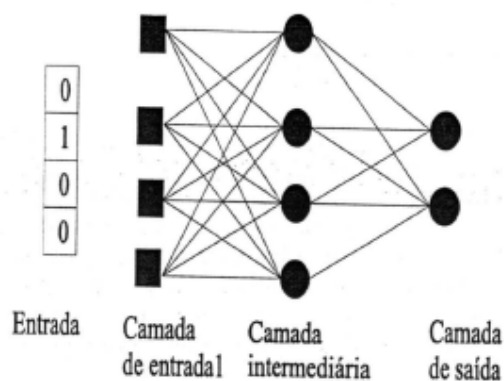


Figura 7 Exemplo de uma rede MLP
Fonte: Braga (et. al, 2000)

2.7 Aprendizado das Redes Neurais

A habilidade de aprender a partir de seu ambiente e de melhorar seu desempenho através da aprendizagem é a propriedade de maior importância para uma rede neural. Uma rede neural aprende através de um processo iterativo chamado regra ou algoritmo de aprendizagem que consiste em correções ou ajustes de seus pesos sinápticos e bias (HAIKIN, 1999).

Consequentemente o processo de aprendizagem é composto dos seguintes eventos:

1. A rede neural é estimulada por um ambiente.
2. A rede neural sofre modificações nos seus parâmetros como resultado dessa estimulação.
3. A rede neural responde de uma maneira nova ao ambiente.

Diversos métodos para treinamento de redes foram desenvolvidos, podendo ser separados em dois paradigmas principais: aprendizado

supervisionado e aprendizado não supervisionado. Outros dois paradigmas bastante conhecidos são os de aprendizado por reforço (que é um caso particular de aprendizado supervisionado) e aprendizado por competição (que é um caso particular de aprendizado não supervisionado).

2.7.1 Aprendizado supervisionado

Este método de aprendizado é o mais utilizado no treinamento das RNAs, é chamado de *aprendizado supervisionado* porque a entrada e saída desejadas para a rede são fornecidas por um supervisor (professor) externo. O objetivo é ajustar os parâmetros da rede, de forma a encontrar uma ligação entre os pares de entrada e saída fornecidos. A Figura 8 ilustra o mecanismo de aprendizado supervisionado.

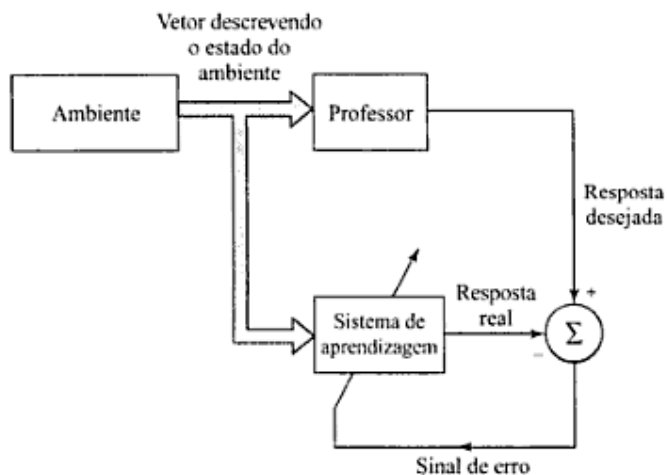


Figura 8 Mecanismo de aprendizado supervisionado

Fonte: Haikin (1999)

A desvantagem da utilização do aprendizado supervisionado é que, a rede não conseguirá aprender novas estratégias para situações não cobertas

pelos exemplos do treinamento da rede na ausência do professor (BRAGA et. al, 2000).

O aprendizado supervisionado pode ocorrer de duas maneiras: *off-line* e *on-line*. No treinamento *off-line*, uma vez obtida uma solução para a rede os dados do conjunto de treinamento não mudam. Caso novos dados sejam adicionados ao conjunto de treinamento, um novo treinamento envolvendo também os dados anteriores deve ser realizado para se evitar interferência no treinamento anterior. Por outro lado, no aprendizado *on-line*, o conjunto de dados muda continuamente, e a rede deve estar em contínuo processo de adaptação (BRAGA et. al, 2000). Os exemplos mais conhecidos de algoritmos para aprendizado supervisionado são a regra delta e a sua generalização para redes de múltiplas camadas, o algoritmo *backpropagation*.

2.7.2 Aprendizado não supervisionado

O aprendizado não supervisionado ao contrário do aprendizado supervisionado, apenas os padrões de entrada são fornecidos a rede, ou seja, não é apresentada a rede nenhuma referência externa fazendo com que este tipo de aprendizado não possua nenhum conhecimento a priori da(s) saída(s) da rede. A Figura 9 abaixo demonstra o mecanismo de aprendizagem não supervisionada.

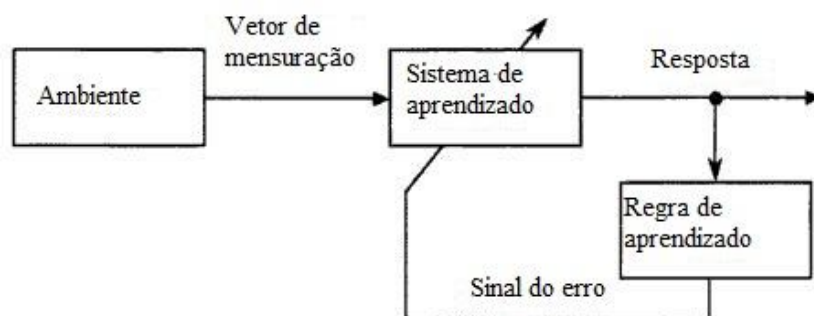


Figura 9 Mecanismo de aprendizado não supervisionado
Fonte: Adaptado de Haikin (1999)

A estrutura do sistema de aprendizado não supervisionado pode adquirir uma variedade de formas diferentes. Ela pode, por exemplo, consistir em uma camada de entrada, uma camada de saída, conexões *feedforward* da entrada para a saída e conexões laterais entre os neurônios da camada de saída. Outro exemplo é uma rede *feedforward* com múltiplas camadas, em que a livre organização procede na base de camada por camada. Nestes dois exemplos, o processo de aprendizado consiste em modificar repetidamente o peso sináptico de todas as conexões do sistema em resposta às entradas. (BRAGA et. al, 2000).

2.7.3 Aprendizado por reforço

A ideia básica por trás do aprendizado por reforço está no papel do professor, ao invés de fornecer a “resposta certa” como no caso do aprendizado supervisionado, no aprendizado por reforço o professor tem um papel de “crítico” no qual ele avalia a resposta (saída) fornecida pela rede. De acordo com Hassoun (1995) as regras do aprendizado por reforço podem ser consideradas como mecanismos estocásticos de busca no qual a probabilidade de reforços externos positivos para um dado conjunto de treinamento é maximizada.

2.8 Regra Delta

A regra delta padrão foi proposta originalmente por Widrow e Hoff (1960) e busca valores dos pesos que minimizam a função custo, nesse caso a soma dos erros quadráticos. De acordo com Haikin (1999, p. 78) a regra delta pode ser formulada como: “o ajuste feito no peso sináptico de um neurônio é proporcional ao produto do sinal do erro pelo sinal de entrada da sinapse em questão.”

A função custo a ser minimizada é apresentada na equação abaixo:

$$E = \frac{1}{2} \sum e^2(k) \quad (2)$$

Onde $e(k)$ representa o erro expressado como:

$$e_k = d_k - o_k \quad (3)$$

Sendo que, d_k representa a saída desejada e o_k a saída calculada. O ajuste de um determinado peso w_k ocorrerá na direção contrária do gradiente da função custo, e a cada passo da iteração será calculado como:

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \quad (4)$$

Os componentes do vetor gradiente podem então ser definidos pelas seguintes equações:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial w_{kj}} \quad (5)$$

Ou

$$\frac{\partial E}{\partial w_{kj}} = -x_j e \quad (6)$$

Logo a Equação (4) de ajuste dos pesos pode ser reescrita como:

$$\Delta w_{kj} = \eta x_j e_k \quad (7)$$

onde η é uma constante de proporcionalidade que define a velocidade com que o vetor de pesos é alterado, sendo também chamado de taxa de aprendizado da rede. Finalmente, a equação genérica para ajuste dos pesos da regra delta, é mostrada abaixo:

$$w(t + 1) = w_k(t) + \eta x(t) \quad (8)$$

2.9 Algoritmo Error Backpropagation

O *backpropagation* é o algoritmo mais conhecido e mais utilizado para o treinamento de rede Multi-Camadas e pode ser definido segundo Braga (et. al, 2000, p 59) como: “um algoritmo supervisionado que utiliza pares (entrada, saída desejada) para, por meio de um mecanismo de correção de erros, ajustar os pesos da rede”. O treinamento utilizando o *backpropagation* ocorre em duas fases, em que cada fase percorre a rede em uma única direção. Estas duas fases são chamadas de fase *forward* e fase *backward* e estão ilustradas nas Figuras 10 e 11 respectivamente. Durante a fase *forward* é feito o cálculo das saídas e seus respectivos erros. A fase *backward*, utiliza o erro calculado durante a fase *forward* para fazer a atualização dos pesos de suas conexões.

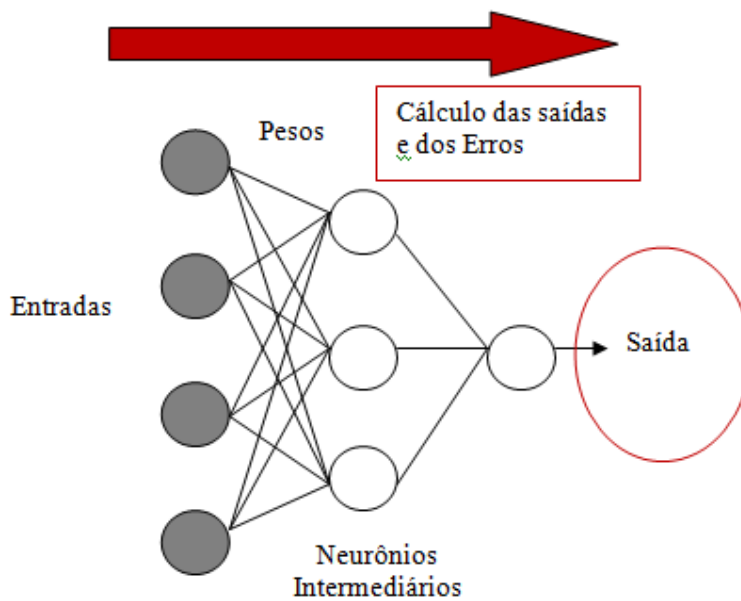


Figura 10 Fase *forward* do algoritmo backpropagation

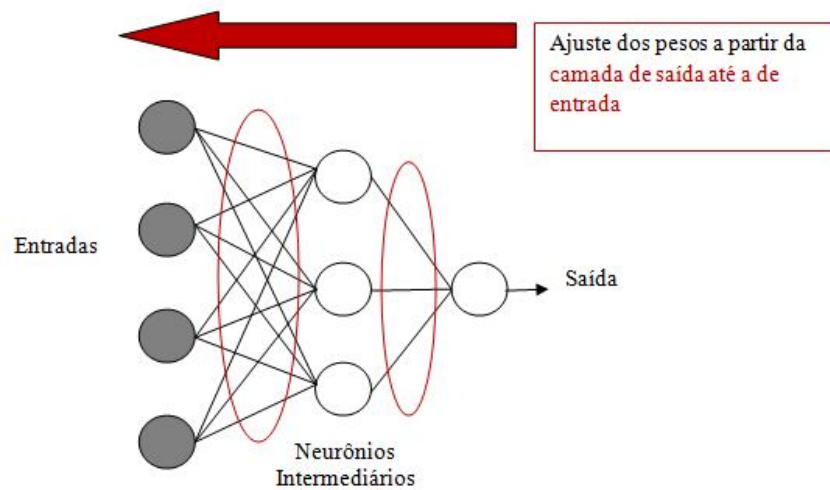


Figura 11 Fase *backward* do algoritmo backpropagation

Braga (et. al, 2000) descreve as duas fases do algoritmo *backpropagation* em:

Fase *forward*:

1. A entrada é apresentada à primeira camada da rede.
2. Para cada camada C_i a partir da camada de entrada
 1. Após os nodos da camada C_i ($i > 0$) calcularem seus sinais de saída, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada C_{i+1} .
3. As saídas produzidas pelos nodos da última camada são comparadas às saídas desejadas.

Nessa fase, são determinadas as saídas para um dado padrão de entrada.

Fase *backward*:

A partir da última camada até a camada de entrada.

1. Os nodos da camada atual ajustam seus pesos de forma a reduzir seus erros.
2. O erro de um nodo das camadas intermediárias é calculado utilizando os erros dos nodos da camada seguinte conectados a ele, ponderados pelos pesos das conexões entre eles.

Nessa fase, as saídas desejadas e calculadas são comparadas e então os pesos ajustados.

O backpropagation trabalha com uma variação da regra delta, apropriada para redes multi-camadas, por isso pode ser considerado como uma regra delta generalizada, na qual a função de custo a ser minimizada é uma função de erro ou energia, definida pela soma dos erros quadráticos, representada pela Equação (9), e o ajuste dos pesos das conexões é definido pela equação Equação (10):

$$E = \frac{1}{2} \sum_p \sum_{k=1}^n (d_k - o_k)^2 \quad (9)$$

$$w_{kj} = w_{kj} + \Delta w_{kj} = w_{kj} + \eta(d_k - o_k)f'(u_k)y_j \quad (10)$$

onde E é a medida de erro total, p é o número de padrões, n é o número de unidades de saída, d_k é a k -ésima saída desejada e o_k ; é a k -ésima saída gerada pela rede. Esta equação define o erro total cometido pela rede, ou a quantidade em que, para todos os padrões p de um dado conjunto, as saídas geradas pela rede diferem das saídas desejadas.

O algoritmo *backpropagation* apresenta uma série de dificuldades ou deficiências que dificultam uma maior disseminação de seu uso, entre eles pode-se destacar:

1. O algoritmo pode convergir para mínimos locais.
2. O tempo de treinamento de redes neurais utilizando *backpropagation*, que tende a ser muito lento. Algumas vezes são necessários milhares de ciclos para se chegar à níveis de erros aceitáveis, principalmente se estiver sendo simulado em computadores.
3. Definir a arquitetura ideal da rede de forma que ela seja tão grande quanto o necessário para conseguir obter as

representações necessárias, e ao mesmo tempo pequena o suficiente para se ter um treinamento mais rápido.

2.10 Dificuldades do treinamento

A aprendizagem pode se tornar cada vez mais difícil à medida que o número de características (variáveis de entrada) de cada caso aumenta, o número de exemplos necessários para se aprender um certo conceito cresce exponencialmente. Este problema é conhecido como maldição da dimensionalidade (*curse of dimensionality*) (VALIANTE, 1984).

Outro grande problema enfrentado no treinamento das redes neurais é o *overfitting*, nesta situação a rede neural perde sua capacidade de generalização e passa a memorizar os exemplos de treinamento. Este problema ocorre geralmente devido ao elevado número de neurônios na camada escondida e também pelo excesso de treinamento. No entanto, se o número de neurônios na camada escondida for muito baixo, ocorrerá o *underfitting*, situação na qual a rede não converge durante seu treinamento.

3 SELEÇÃO DE CARACTERÍSTICAS

No aprendizado de máquina e estatística seleção de características, também conhecido como seleção de variáveis, é uma técnica que consiste na seleção de um subconjunto das características mais relevantes de um determinado conjunto de dados.

Esta seção descreve alguns, dos vários métodos de seleção de características utilizando RNAs encontradas na literatura.

3.1 Algoritmo de Garson

O algoritmo original proposto por Garson (1991) envolve essencialmente o particionamento dos pesos das conexões entre a camada escondida e a de saída de cada neurônio intermediário em componentes associados com cada neurônio de entrada.

Para a obtenção da importância relativa de cada variável o algoritmo executa os seguintes passos:

1. Para cada neurônio intermediário i , o valor absoluto do peso da conexão entre este neurônio e um de saída é multiplicado pelo valor absoluto do peso da conexão entre o mesmo neurônio escondido e um neurônio de entrada. Este cálculo deve ser feito para todos os j -ésimos neurônios da camada de entrada. Então o produto P_{ij} é obtido através da Equação (11):

$$P_{ij} = w_{ij} \times w_{io} \quad (11)$$

2. Para cada neurônio escondido, divide-se P_{ij} pela soma de todos os P_{ij} para cada neurônio de entrada, obtendo Q_{ij} . Assim:

$$Q_{ij} = P_{ij} / \sum_{j=1}^n P_{ij} \quad (12)$$

3. Para cada neurônio de entrada, os valores de Q_{ij} são somados obtendo-se S_j :

$$S_j = \sum_{i=1}^n Q_{ij} \quad (13)$$

4. Dividindo-se cada valor de S_j pela soma de todos os valores de S_j , obtemos a importância relativa R para cada variável, j :

$$R_j = \left(\frac{S_j}{\sum_{j=1}^n S_j} \right) \times 100 \quad (14)$$

O algoritmo de Garson utiliza valores absolutos dos pesos das conexões para o cálculo da contribuição da variável, não permitindo uma análise da direção das modificações ocorridas na variável de saída quando ocorre alteração nas variáveis de entrada. (VALENÇA, 2007).

3.2 Análise de Sensibilidade

Este método proposto por Valença (2007) baseia-se nas derivadas parciais da variável de saída com relação aos pesos nas conexões, isto é, o método utiliza os resultados obtidos das derivadas parciais (sensibilidade) durante o treinamento da rede com o algoritmo *backpropagation*.

A sensibilidade das variáveis de entrada é calculada através da retropropagação do erro pelo algoritmo *backpropagation*. Primeiro se realiza o cálculo do erro para cada neurônio da camada de saída (tem-se o valor calculado e a resposta desejada).

$$e_k = d_k - o_k \quad (15)$$

A sensibilidade de cada variável é calculada para cada exemplo T (T varia de 1 até n onde n é o número total de exemplos de treinamento) de acordo com a seguinte equação:

$$Sen_{jC} = \sum_{k=1}^{N_{hid}} w_{kj} f'(net_k) \times \sum_{i=1}^{N_{out}} w_{ik} f'(net_i) e_i \quad (16)$$

onde: Sen_{jC} é a sensibilidade para cada variável j de entrada em relação a saída para um dado exemplo de treinamento C , w_{kj} são os pesos sinápticos das conexões entre os neurônios escondidos e a camada de entrada, $f'(net_k)$ é a derivada da função de ativação dos neurônios da camada escondida, N_{hid} é o número total de neurônios na camada escondida, w_{ik} são os pesos sinápticos das conexões entre os neurônios da camada escondida e os da camada de saída e $f'(net_i)$ é a derivada da função de ativação dos neurônios da camada de saída.

A partir da Sen_{jC} a contribuição percentual de cada variável j de entrada com relação a variável de saída é calculada por:

$$cont_j = \frac{\sum_{C=1}^N Sen_{jC}^2}{\sum_{j=1}^{N_{imp}} \sum_{C=1}^N Sen_{jC}^2} \quad (17)$$

onde: N representa o número total de exemplos de treinamento e N_{imp} é o número de entradas (neurônios na camada de entrada).

De acordo com Valença (2007), os resultados de sensibilidade e de contribuição obtidos por este método permitem que sejam realizadas duas análises: Uma análise da variação da saída para pequenas mudanças nas

variáveis de entrada e uma classificação da importância de cada uma das variáveis de entrada com relação a variável de saída da rede.

3.3 Método *PaD*

Este método, assim como o anterior, baseia-se em derivadas parciais, porém este método calcula a derivada parcial da saída da rede em relação à entrada. O perfil das variações da saída para pequenas alterações de uma variável de entrada é obtido através do cálculo da derivada parcial da saída da RNA em relação à entrada. (DIMOPOULOS et al., 1999).

Para uma rede com n_i entradas, uma camada escondida com n_h neurônios e uma saída, a derivada parcial da saída y_j em relação à entrada x_j ($j = 1, \dots, N$, onde N é o total de amostras) é calculada de acordo com a equação a seguir:

$$d_{ji} = f'(net_j) \sum_{h=1}^{n_h} w_{ho} f'(net_h) w_{ih} \quad (18)$$

onde $f'(net_j)$ e $f'(net_h)$ são as derivadas das funções de ativação dos neurônios da camada escondida e da de saída, respectivamente.

w_{ho} e w_{ih} são os pesos entre o neurônio de saída e o h -ésimo neurônio escondido, e entre o i -ésimo neurônio de entrada e o h -ésimo neurônio escondido.

A contribuição relativa de uma entrada para a saída da RNA é calculada através da soma dos quadrados das derivadas parciais obtida pela seguinte equação:

$$SSD_i = \sum_{j=1}^N (d_{ji})^2 \quad (19)$$

Um valor SSD (Sum of Square Derivatives) é obtido para cada variável i de entrada, e a que apresentar o maior valor consequentemente será a que mais exerce influência na determinação da saída.

3.4 Método Perturb

Este método tem como objetivo avaliar o efeito de pequenas mudanças em cada variável de entrada na saída da rede neural. O algoritmo ajusta os valores de entrada de uma variável, enquanto mantém todas as outras inalteradas. As respostas da variável de saída em relação a cada mudança na variável de entrada são armazenadas.

A variável de entrada cujas mudanças afetam mais a saída, será a que exerce maior influência relativa no modelo. Na verdade, é esperado que o erro médio quadrático (EMQ) da saída da rede neural aumente à medida que uma maior quantidade de ruído é adicionado à variável de entrada selecionada (YAO et al., 1998).

As mudanças nas variáveis, Equação (20), ocorrem geralmente adicionando à variável selecionada de 10% a 50% (valores mais utilizados) de seu valor (GREVEY et al., 2003).

$$x_i = x_i + \delta \quad (20)$$

Onde x_i é a variável de entrada selecionada e δ é a mudança (10% a 50%) a ser adicionado a x_i .

3.5 Correlação

A correlação linear mede a “força” ou “grau” de relacionamento entre duas variáveis.

A medida da correlação, Equação (21), denominada coeficiente de correlação varia no intervalo de -1 a 1. Sendo que -1 indica correlação negativa perfeita, 0 ausência de correlação e 1 correlação positiva perfeita.

$$r = \frac{n(\sum xy) - (\sum x) \times (\sum y)}{\sqrt{n(\sum x^2) - (\sum x)^2} \times \sqrt{n(\sum y^2) - (\sum y)^2}} \quad (21)$$

Onde r é o coeficiente de correlação, x e y são as variáveis que se deseja medir a correlação e n é o total de amostras. O coeficiente quanto mais próximo de 1 ou -1 indica que a variável exerce mais influência na determinação da saída da rede.

4 METODOLOGIA

Este capítulo descreve os materiais e métodos utilizados neste trabalho, o processo de desenvolvimento e aplicação das teorias apresentadas nas Seções 2 e 3.

4.1 Ambiente de Programação (Scilab)

Para a implementação das RNAs utilizadas neste trabalho será utilizado o ambiente de programação Scilab 5.3.3 (Scilab Consortium). O Scilab é um software livre distribuído sobre CeCILL license, compatível com a General Public License (GPL) para computação numérica semelhante ao Matlab e conta com linguagem de programação própria de alto nível proporcionando um ambiente de computação poderosa para aplicações de engenharia e científicas.

O Scilab foi desenvolvido em 1990 pelos pesquisadores do Institut National de Recherche en Informatique et en Automatique (INRIA) e do École Nationale des Ponts et Chaussées (ENPC). A partir de maio 2003 seu controle passou a fazer parte do Consórcio Scilab no qual fazem partes várias indústrias e instituições acadêmicas. O Scilab é atualmente usado em diversos ambientes industriais e educacionais pelo mundo.

O Scilab por ser um software livre ainda conta com uma série de módulos desenvolvidos por usuários e outras empresas, aumentando ainda mais sua gama de aplicações. Neste trabalho em específico será utilizado o módulo *ANNToolbox* versão 0.5.2, desenvolvido por Hristev (2011), que contém todas as funções necessárias para a implementação e treinamento de redes neurais.

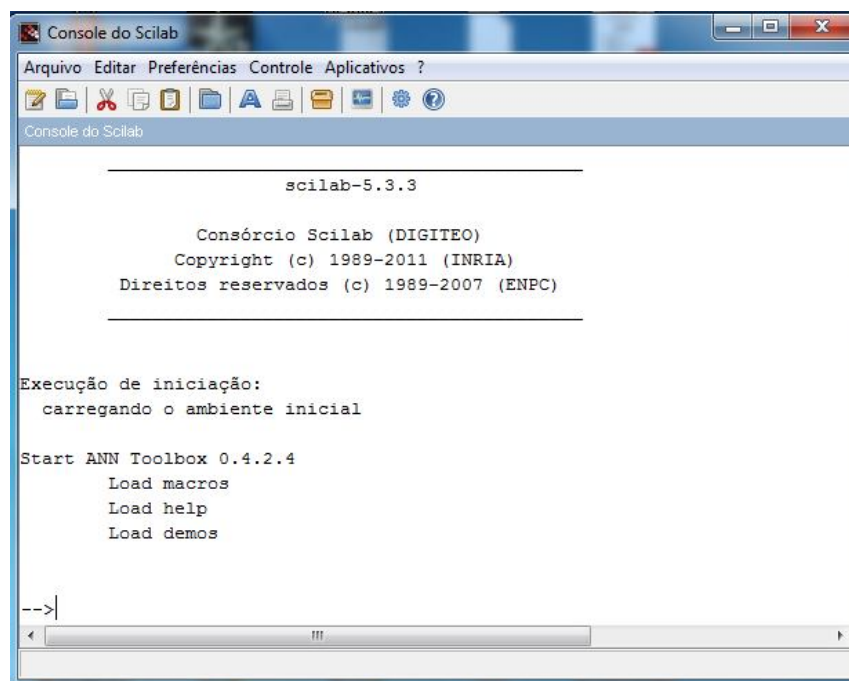


Figura 12 Tela inicial do Scilab

4.2 Treinamento das Redes Neurais

Esta seção descreve os passos e procedimentos a serem adotados antes e durante o treinamento das redes neurais.

4.2.1 Pré-processamento dos dados

Os dados utilizados para o treinamento das redes neurais foram obtidos do repositório da *UCI Machine Learning* (FRANK et al., 2011).

Normalmente os dados de entrada estão em intervalos de variação bastante distintos, isto faz com que seu treinamento e aprendizagem fiquem prejudicados, pois a rede neural pode interpretar valores mais altos como de maior importância e valores menores como menos importantes.

Para contornar este problema é utilizada a técnica de normalização dos dados, isto faz com que todos os atributos fiquem com valores no

intervalo da função de ativação dos neurônios, no caso deste trabalho o Scilab adota como padrão a função de ativação sigmoidal que possui o intervalo variando de 0 a 1. Os dados serão normalizados de acordo com a seguinte equação:

$$y = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (22)$$

onde y é o valor normalizado; x_i é o valor original; x_{min} e x_{max} são, respectivamente, o valor mínimo e o valor máximo do conjunto de dados para determinado atributo.

Além do problema de grandes variações no intervalo dos dados, alguns atributos encontram-se com valores nominais o que torna seu processamento pela rede impossível de ser executado. A técnica mais utilizada para resolver este problema é a binarização, que consiste em atribuir números binários aos dados com valores nominais.

Os dados obtidos do repositório da UCI Machine Learning serão separados aleatoriamente em duas categorias: dados de treinamento (80%), que serão utilizados para o treinamento da rede e dados de teste (20%), que serão utilizados para verificar seu desempenho sob condições reais de utilização.

4.2.2 Configuração da rede

As redes neurais utilizadas neste trabalho são do tipo MLP com apenas uma camada escondida, função de ativação sigmoidal. O algoritmo utilizado para o treinamento foi *Error Backpropagation* possuindo o número épocas como critério de parada. Para otimizar o treinamento alguns parâmetros serão inicializados manualmente com valores próximos de zero como no caso da taxa de aprendizagem, já os demais parâmetros serão

inicializados automaticamente pelas funções implementadas pelo toolbox de redes neurais. Para as redes implementadas até o momento os valores para a taxa de aprendizado de 0.1548 e momentum de 0.238 foram os que apresentaram melhores resultados no treinamento.

Para simplificar a implementação dos métodos de seleção de características será utilizada apenas uma camada escondida. A Equação (23) serve como um parâmetro para a definição do número total de neurônios na camada escondida.

$$N_{hid} = \frac{N_{out} + N_{inp}}{2} \quad (23)$$

Onde N_{hid} é o número de neurônios na camada escondida, N_{out} é o número de neurônios na camada de saída, N_{inp} é o número de entradas da rede neural. Como citado anteriormente essa equação serve apenas como um parâmetro para a definição do número de neurônios na camada escondida, este número pode ser ajustado para mais ou para menos dependendo dos resultados e do problema em questão.

Com o intuito de se obter resultados mais consistentes, para cada problema serão treinadas dez redes neurais, executando-se em seguida os algoritmos de seleção de características em cada uma delas. Uma vez executados os algoritmos, será calculada a média e desvio padrão dos resultados de cada método.

4.3 Base de dados

Esta seção descreve os problemas e os dados a serem utilizados nos treinamentos das redes neurais e na extração da importância de cada atributo dos dados.

4.3.1 Performance Relativa da CPU

O objetivo deste problema é determinar a performance relativa (PRP) da CPU tendo como base algumas variáveis de entrada descritas abaixo:

1. vendor name: 30 (adviser, amdahl, apollo, basf, bti, burroughs, c.r.d, cambex, cdc, dec, dg, formation, four-phase, gould, honeywell, hp, ibm, ipl, magnuson, microdata, nas, ncr, nixdorf, perkin-elmer, prime, siemens, sperry, sratus, wang)
2. Model Name: many unique symbols
3. MYCT: machine cycle time in nanoseconds
4. MMIN: minimum main memory in kilobytes
5. MMAX: maximum main memory in kilobytes
6. CACH: cache memory in kilobytes
7. CHMIN: minimum channels in units
8. CHMAX: maximum channels in units
9. PRP: published relative performance

As variáveis 1 e 2, por apenas conter nomes de marcas e modelos foram descartadas no treinamento. A base de dados para este problema conta com 209 exemplos ao todo e pode ser encontrada no repositório da UCI *Machine Learning* (FRANK et al., 2011).

4.3.2 Iris

Originalmente o objetivo deste problema é fazer a classificação de algumas espécies da planta Iris (*Iris-setosa*, *Iris-versicolor* e *Iris-virginica*) levando em consideração os seguintes atributos:

1. Comprimento das sépalas (cm)
2. Largura das sépalas (cm)
3. Comprimento das pétalas (cm)
4. Largura das pétalas (cm)

Como alguns métodos de seleção de características restringem a rede neural a possuir apenas uma saída como o de Garson, por exemplo, o problema foi modificado para classificar apenas duas espécies (*Iris-setosa*, *Iris-versicolor*).

A base de dados para este problema conta com 100 exemplos ao todo sendo que 50% são da espécie *Iris-setosa* e 50% da espécie *Iris-versicolor*. Essa base de dados pode ser encontrada no repositório da UCI *Machine Learning* (FRANK et al., 2011).

4.3.3 Resistência do concreto

A resistência do concreto é uma função altamente não linear da idade e componentes. Então o objetivo deste problema é estimar a resistência do concreto à compressão levando em conta a idade e os componentes especificados abaixo:

1. Cimento kg /m³
2. Resíduos do alto forno kg/m³
3. Fly Ash (resíduo da combustão) kg/m³
4. Água kg/m³
5. Superplastificante kg/m³
6. Agregado graúdo kg/m³ (todo o agregado que fica retido na peneira de número 4, malha quadrada com 4,8 mm de lado)
7. Agregado miúdo kg/m³ (todo agregado que consegue passar por esta peneira.)

8. Idade 1 a 365 dias.

A base de dados para este problema conta com 1030 exemplos ao todo e poder ser encontrada no repositório da *UCI Machine Learning* (FRANK et al., 2011).

4.4 Análise dos resultados

Com o intuito de se obter resultados mais consistentes, para cada problema serão treinadas dez redes neurais, executando-se em seguida os algoritmos de seleção de características em cada uma delas. Uma vez executados os algoritmos, será calculada a média e desvio padrão dos resultados de cada método.

Após o treinamento das redes neurais e execução dos algoritmos de seleção de características, será feita uma análise comparativa dos resultados confrontando a importância das variáveis de entrada estipuladas por cada método.

Para determinar qual método foi mais eficiente na seleção das variáveis de entrada será feito um novo treinamento das redes neurais apenas com entradas consideradas mais importantes de cada método. Espera-se que após a eliminação destas variáveis a rede neural tenha um desempenho (MSE) semelhante ao que ela conseguiu com todos os atributos. Então será considerado o método mais eficiente aquele que selecionar as variáveis que apresentem após um novo treinamento um erro médio quadrático (MSE) mais próximo ou menor que o MSE original com a presença de todas variáveis.

5 RESULTADOS

Nesta seção será feita a apresentação e análise comparativa dos resultados, confrontando a importância das variáveis de entrada estipuladas por cada método após o treinamento das redes neurais e execução dos algoritmos de seleção de características.

5.1 Iris

Esta seção demonstra os resultados obtidos pelos métodos de seleção de características para o problema da Iris e também os resultados do retreinamento das Redes Neurais com as variáveis selecionadas por cada método.

5.1.1 Definição da importância das variáveis de entrada

Para o problema da Iris as redes neurais apresentaram um erro de treinamento de 0,009582 com desvio padrão de 0,001639 e um erro de teste de 0,001109 e desvio padrão de 0,000213 conforme o gráfico abaixo.

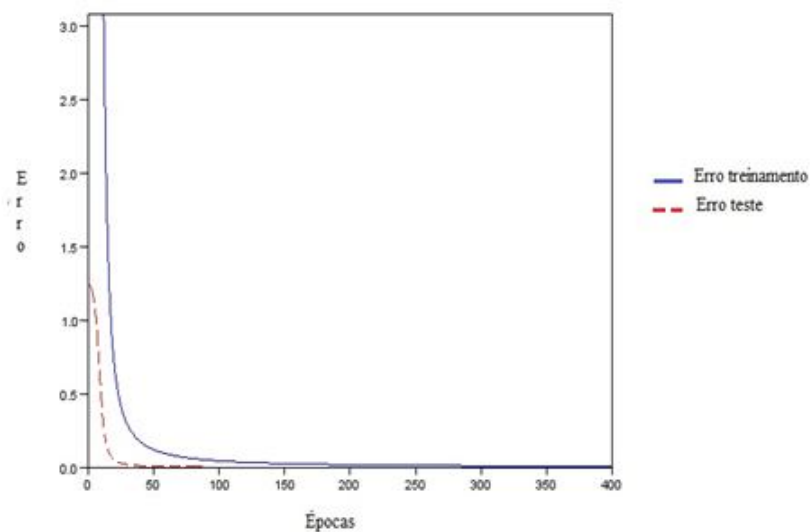


Gráfico 1 Erro médio quadrático para os dados de treinamento e teste em função do número de épocas

Após o treinamento das redes neurais os métodos de seleção de características foram executados apresentando os resultados exibidos na tabela abaixo:

Tabela 1 - Importância das variáveis de acordo com cada método de seleção de características para o problema da Iris.

	Garson	Correlação	Perturb	Sensibilidade	PaD
v1	11,3%	0,7283	0,00952	$3,597 \times 10^{-5}$	0,064679
v2	23,7%	- 0,684	0,01085	$17,971 \times 10^{-5}$	0,234103
v3	31,6%	0,97	0,01370	$33,394 \times 10^{-5}$	0,419266
v4	29,2%	0,9602	0,01701	$30,75 \times 10^{-5}$	0,407717

Para o problema da Iris todos os métodos apontaram as variáveis **v3** e **v4** (Comprimento das pétalas e Largura das pétalas) como as mais importantes, já a menos importante foi a variável **v2** (Largura das sépalas) de acordo com a Correlação e **v1** (Comprimento das sépalas) para os demais métodos.

A comparação entre estes métodos é melhor exemplificada pelos Gráficos logo abaixo.

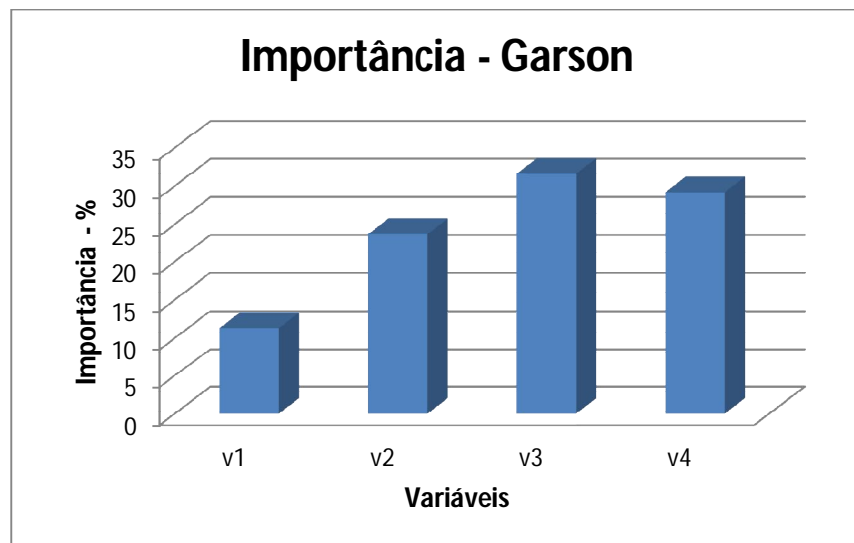


Gráfico 2 Importância das variáveis de acordo com o algoritmo de Garson

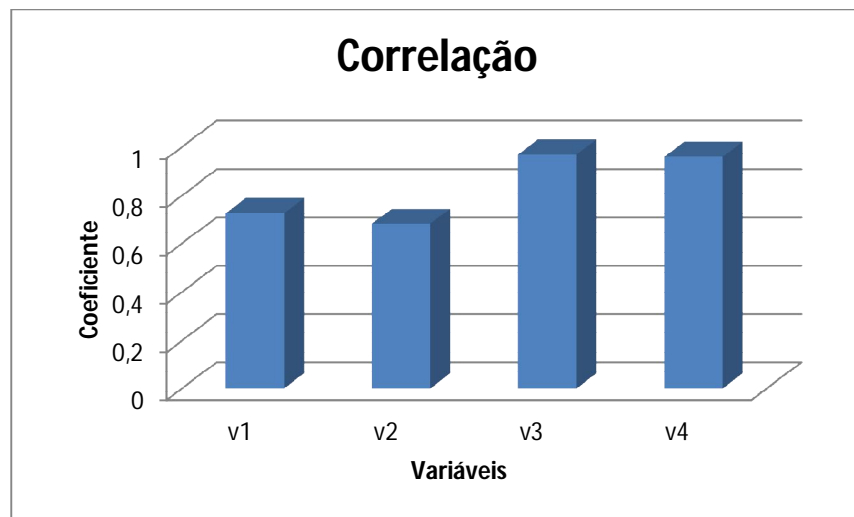


Gráfico 3 Correlação entre as variáveis do problema Iris

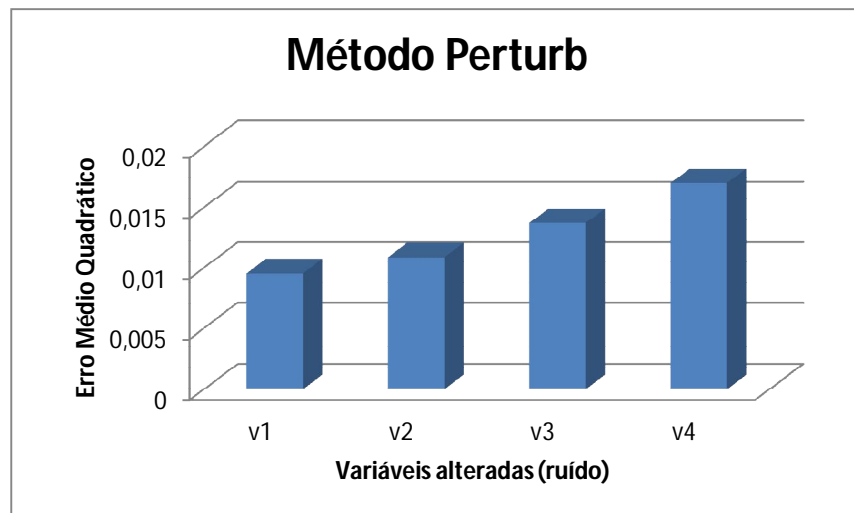


Gráfico 4 Erro médio quadrático em função da adição de ruído nas variáveis de entrada

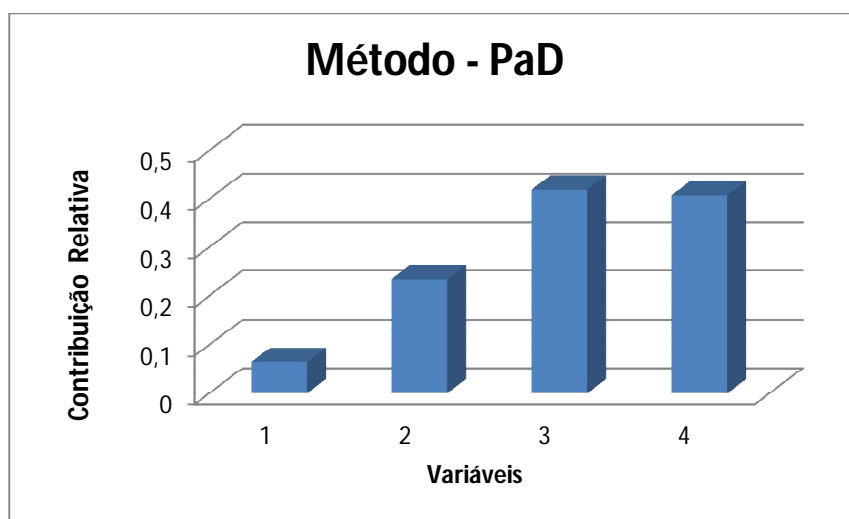


Gráfico 5 Contribuição Relativa de cada variável de acordo com o método PaD

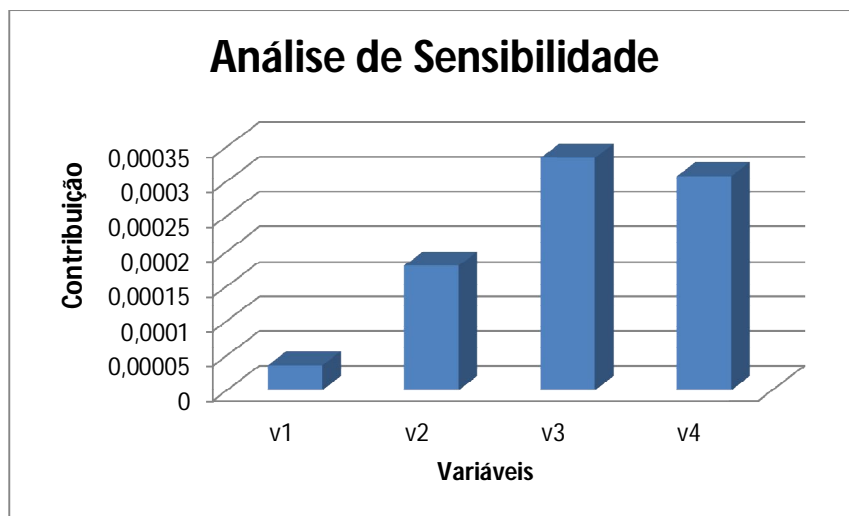


Gráfico 6 Contribuição de cada variável de acordo com o método de Análise de Sensibilidade

5.1.2 Retreinamento da RNA

Como todos os métodos apresentaram resultados semelhantes na determinação das variáveis mais importantes, neste caso **v3** e **v4** (Comprimento das pétalas e Largura das pétalas), para critério de avaliação

dos métodos as redes foram retreinadas excluindo-se a variável considerada menos significativa, **v2** (Largura das sépalas) para o método de correlação e **v1** (Comprimento das sépalas) para os demais.

Excluindo a variável **v1**, as redes neurais apresentaram um erro médio quadrático de 0,010262 para os dados de treinamento. Comparando o novo erro com o original percebe-se que o mesmo sofre um pequeno aumento em relação ao erro de treinamento original.

Excluindo a variável **v2** obteve-se um erro médio quadrático de 0,01352 para os dados de treinamento. Comparando com o erro original e também com os erros obtidos no treinamento sem a variável **v1**, nota-se que o erro de teste foi significativamente maior, o que indica um menor desempenho do método de correlação em relação aos demais.

5.2 CPU Performance

Esta seção demonstra os resultados obtidos pelos métodos de seleção de características para o problema da CPU Performance e também os resultados do retreinamento das Redes Neurais com as variáveis selecionadas por cada método.

5.2.1 Definição da importância das variáveis de entrada

Para o problema CPU Performance as redes neurais apresentaram um erro de treinamento de 0,15200 com desvio padrão de 0,045741 e um erro de teste de 0,052374 e desvio padrão de 0,045741 conforme o gráfico 7.

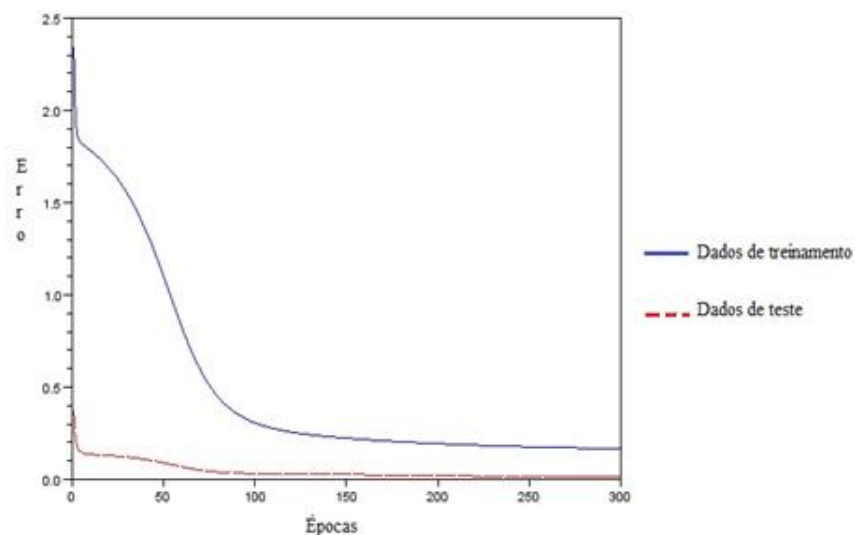


Gráfico 7 Erro médio quadrático para os dados de treinamento e teste em função do número de épocas

Após o treinamento das redes neurais os métodos de seleção de características foram executados apresentando resultados exibidos na tabela abaixo:

Tabela 2 - Importância das variáveis de acordo com cada método de seleção de características para o problema CPU Performance.

	Garson	Correlação	Perturb	Sensibilidade	PaD
V1	14,9%	-0.3071	0,01855	0,005037	1,250068
V2	20,5%	0.7949	0,01502	0,013737	2,79839
V3	28,9%	0.8630	0,01592	0,023963	5,923316
V4	13,4%	0.6626	0,01761	0,004184	0,886935
V5	9,3%	0.6089	0,01402	0,001002	0,280499
V6	13%	0.6052	0,01689	0,004199	1,254882

Para o problema da CPU Performance a variável mais importante de acordo com todos os métodos foi **v3** (maximum main memory in kilobytes). A variável considerada menos importante pelo algoritmo de Garson foi **v5** (minimum channels in units), **v6** (maximum channels in units) para o método Perturb, **v1**(machine cycle time in nanoseconds) pelo método de correlação e **v4** (minimum main memory in kilobytes) para os métodos PaD e Análise de Sensibilidade.

A comparação entre estes métodos é melhor exemplificada pelos gráficos abaixo.

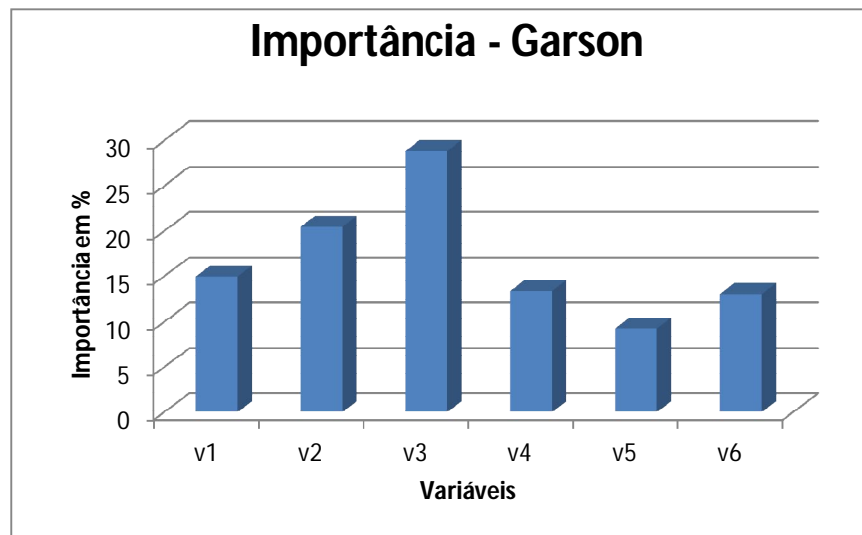


Gráfico 8 Importância das variáveis de acordo com o algoritmo de Garson

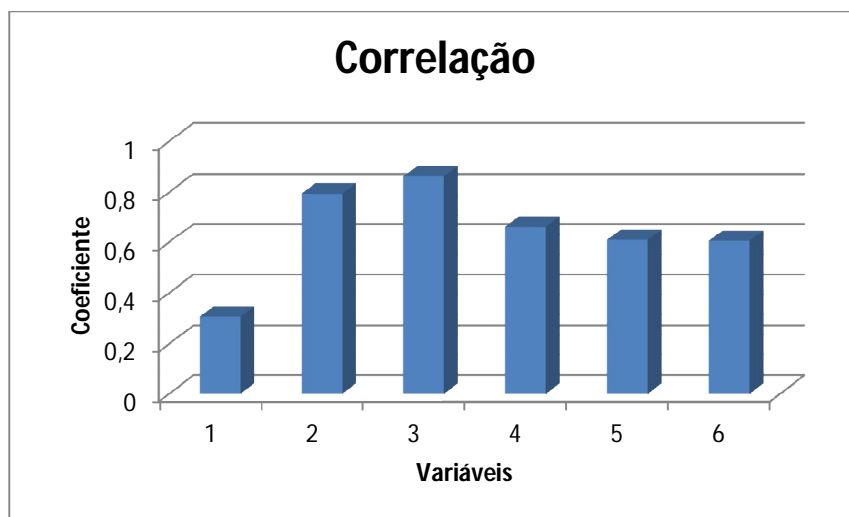


Gráfico 9 Correlação entre as variáveis

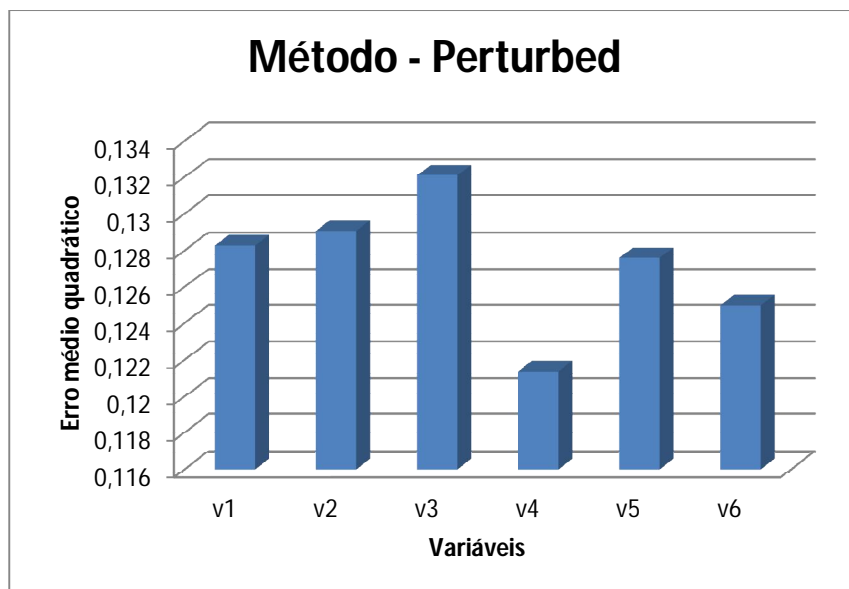


Gráfico 10 Erro médio quadrático em função da adição de ruído nas variáveis de entrada

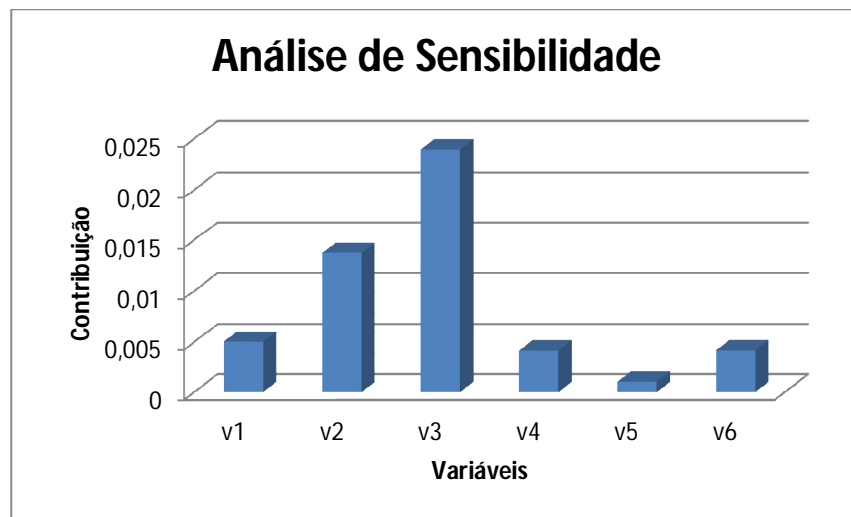


Gráfico 11 Contribuição de cada variável de acordo com o método de Análise de Sensibilidade

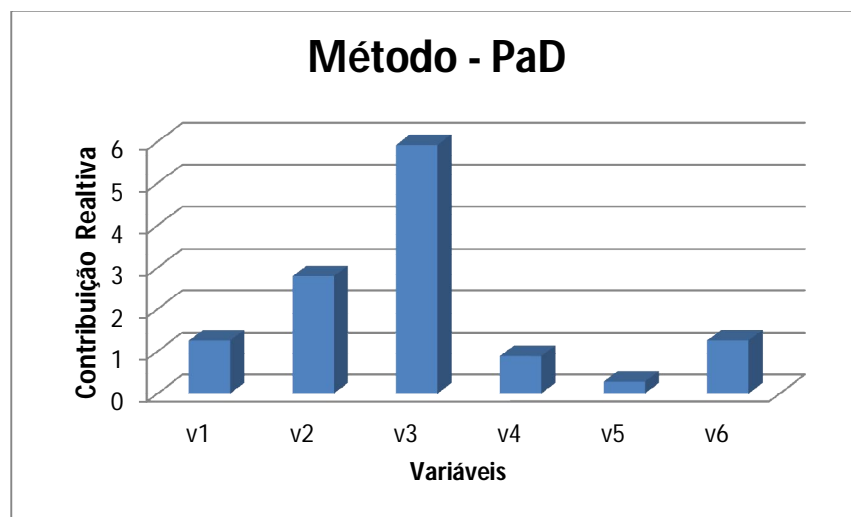


Gráfico 12 Contribuição Relativa de cada variável de acordo com o método de PaD

5.2.2 Retreinamento da RNA

Para o problema da CPU Performance o retreinamento das redes foi realizado excluindo-se as duas variáveis menos significativas estipuladas por cada método:

- Garson: **v5** (minimum channels in units) e **v6** (maximum channels in units).
- Correlação: **v1** (machine cycle time in nanoseconds) e **v6**.
- Perturb: **v4** (cache memory in kilobytes) e **v6**.
- PaD e Análise de Sensibilidade: **v4** e **v5**.

Os resultados do retreinamento são exibidos na tabela abaixo:

Tabela 3 – Resultado do retreinamento da RNA – CPU Performance.

	Garson	Correlação	Perturb	Sensibilidade e PaD
Variáveis excluídas	v5 e v6	v1 e v6	v4 e v6	v4 e v5
EMQ original	0,1520	0,1520	0,1520	0,1520
EMQ retreinamento	0,1677	0,1573	0,1789	0,1540

Excluindo as variáveis **v5** e **v6** obteve-se um novo erro médio quadrático de 0,16774 para os dados de treinamento. Confrontando o novo erro de treinamento obtido com o original nota-se um aumento quase que insignificante.

Removendo as variáveis **v1** e **v6** as redes neurais apresentaram um novo erro médio quadrático de 0,1573 para os dados de treinamento, valor praticamente igual se comparado ao original.

Excluindo-se **v4** e **v6** o erro de treinamento obtido foi de 0,17899, apresentando um significativo aumento em relação ao erro médio quadrático original. E por fim, excluindo **v4** e **v5** o novo erro obtido foi de 0,15407, valor bem próximo ao original.

5.3 Resistência do Concreto

Esta seção demonstra os resultados obtidos pelos métodos de seleção de características para o problema da Resistência do Concreto e também os resultados do retreinamento das Redes Neurais com as variáveis selecionadas por cada método.

5.3.1 Definição da importância das variáveis de entrada

Para o problema da Resistência do concreto as redes neurais apresentaram um erro de treinamento de 0,65306 com desvio padrão de 0,045741 e um erro de teste de 0,84274 e desvio padrão de 0,05041 conforme, ao gráfico a seguir.

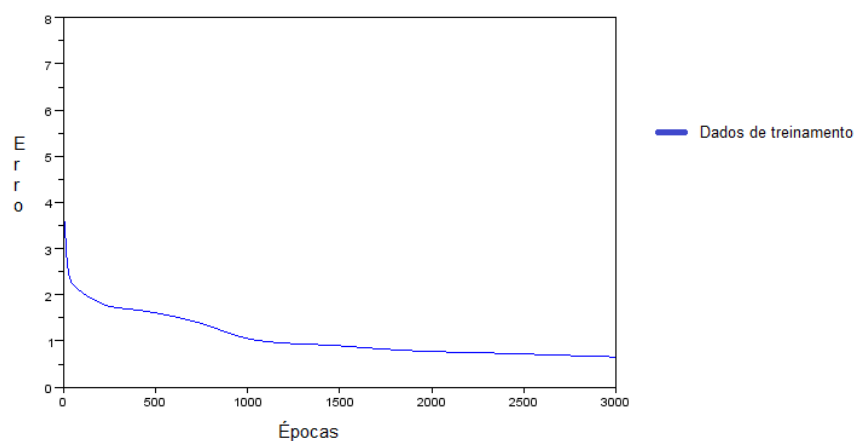


Gráfico 13 Erro médio quadrático para os dados de treinamento função do número de épocas

Após o treinamento das redes os métodos foram executados apresentando os resultados exibidos na tabela abaixo:

Tabela 4 - Importância das variáveis de acordo com cada método de seleção de características para o problema da Resistência do Concreto.

	Garson	Correlação	Perturb	Sensibilidade	PaD
V1	15,7 %	0,4978	0,70970	10,3901	2705,83
V2	12 %	0,1348	1,06426	2,46488	778,526
V3	15,2 %	-0,10575	1,43949	5,25799	1711,86
V4	9 %	-0,28963	0,64357	2,97607	1299,15
V5	13,2%	0,36607	1,01091	5,11057	1913,59
V6	9,8 %	-0,16493	0,88616	0,95426	316,85
V7	15,4%	-0,16724	0,99535	6,20653	2462,41
V8	13%	0,328873	0,81137	12,468	3104,60

Para o problema da Resistência do Concreto a variável mais importante de acordo com o algoritmo de Garson, o Método de Correlação, PaD e Análise de Sensibilidade, foi a **v1** (Cimento), já para o método Perturb a variável mais significativa foi a **v3** (resíduo da combustão). A variável considerada menos importante pelo algoritmo de Garson e o método Perturb foi a **v4** (Água), **v3**(resíduo da combustão) pelo método de correlação e **v6** (Agregado graúdo) pela Análise de Sensibilidade e PaD.

A comparação entre estes métodos é melhor exemplificada pelos gráficos abaixo.

Importância - Garson

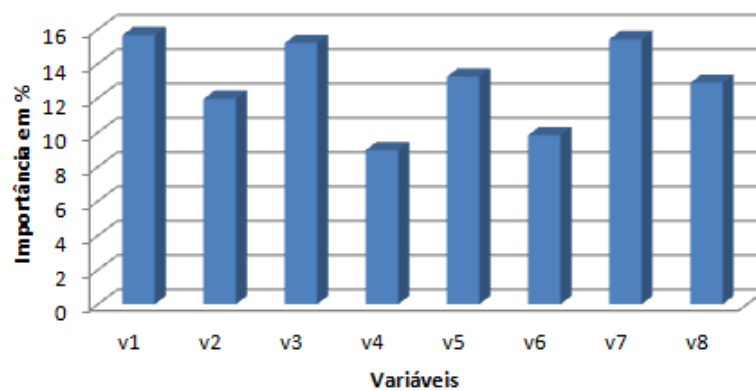


Gráfico 14 Importância das variáveis de acordo com o algoritmo de Garson

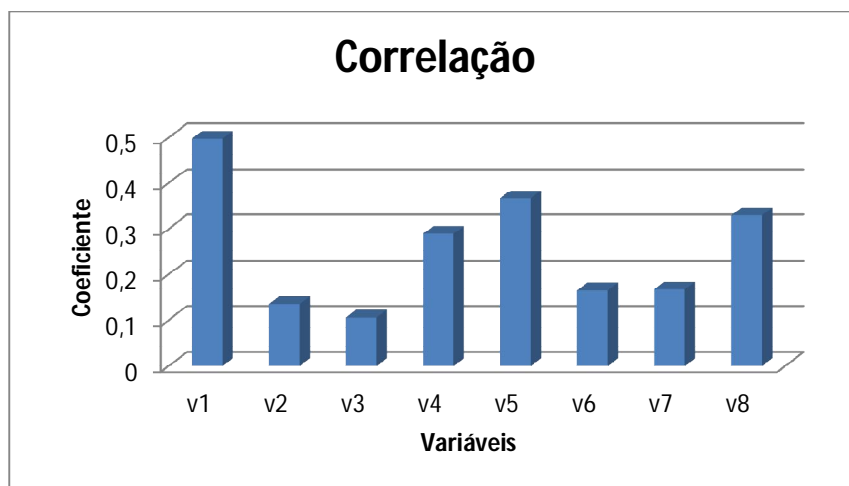


Gráfico 15 Correlação entre as variáveis

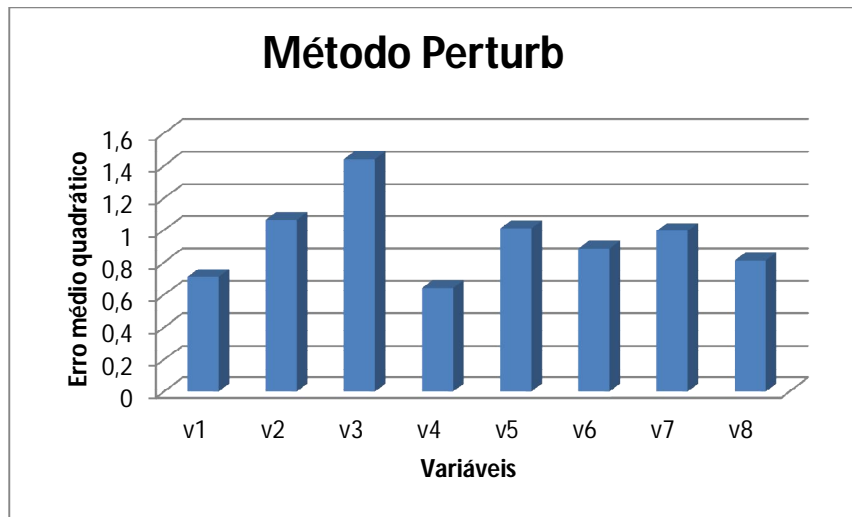


Gráfico 16 Erro médio quadrático em função da adição de ruído nas variáveis de entrada

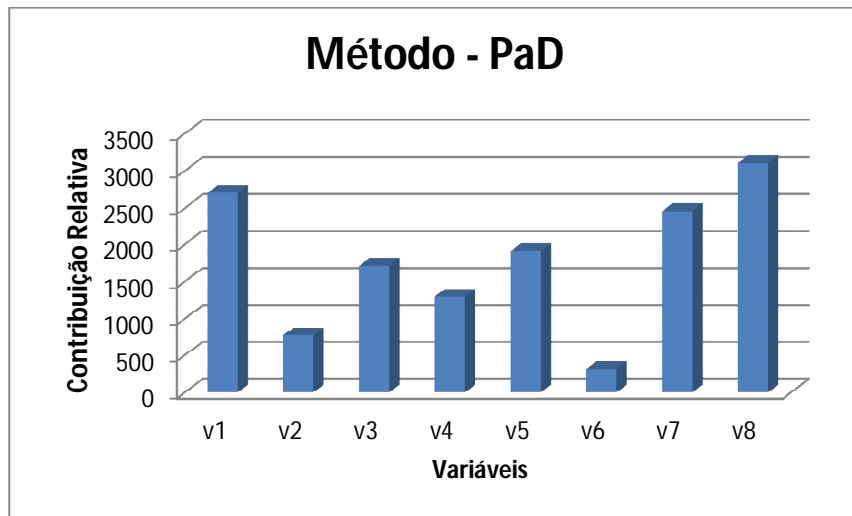


Gráfico 17 Contribuição Relativa de cada variável de acordo com o método de PaD

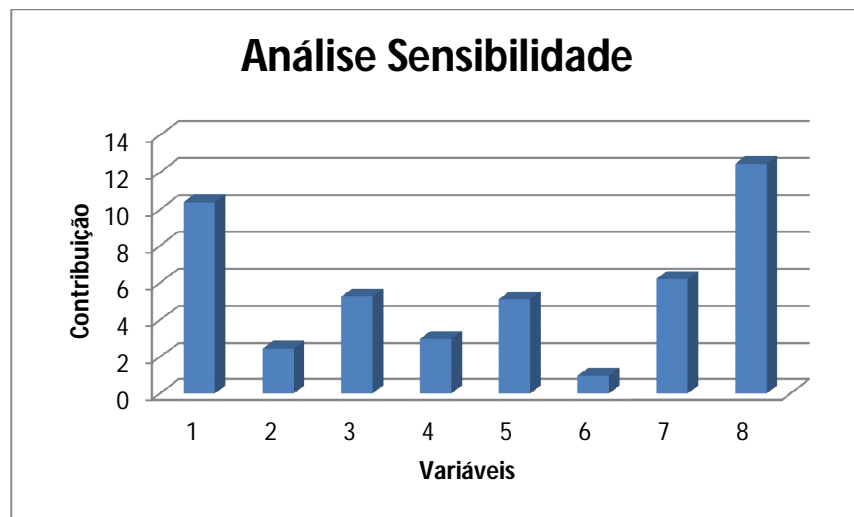


Gráfico 18 Contribuição de cada variável de acordo com o método de Análise de Sensibilidade

5.3.2 Retreinamento da RNA

Para o problema da Resistência do Concreto o retreinamento das redes foi realizado excluindo as duas variáveis menos significativas estipuladas por cada método:

- Garson: **v4** (Água) e **v6** (Agregado graúdo).
- Correlação: **v2** (Resíduos do alto forno) e **v3** (Fly Ash).
- Perturb: **v1** (Cimento) e **v4**
- Pad e Análise Sensibilidade: **v2** e **v6**.

Os resultados do retreinamento são exibidos na tabela abaixo:

Tabela 5 – Resultado do retreinamento da RNA – Resistência Concreto

	Garson	Correlação	Perturb	Sensibilidade e PaD
Variáveis excluídas	v4 e v6	v2 e v3	v1 e v4	v2 e v6
EMQ original	0,6530	0,6530	0,6530	0,6530
EMQ retreinamento	1,0721	1,3764	1,3346	1,0247

Excluindo as variáveis **v4** e **v6** obteve-se um erro médio quadrático de 1,0721 para os dados de treinamento, o que representa um aumento significativo se comparado ao erro médio quadrático original.

Removendo as variáveis **v2** e **v3** as redes neurais apresentaram um erro médio quadrático de 1,376448 para os dados de treinamento, valor relativamente alto se comparado ao erro original.

Excluindo as variáveis **v1** e **v4** o erro de treinamento obtido foi de 1,334614, apresentando um significativo aumento em relação ao erro médio quadrático original. E por fim eliminando-se **v2** e **v6** obteve-se um novo erro de 1,0247.

6. CONCLUSÃO

O principal objetivo deste trabalho foi implementar os métodos de seleção de características de dados em Redes Neurais Artificiais encontrados na literatura, visando identificar a contribuição de cada variável de entrada em relação a saída. O uso de algoritmos baseados nos pesos e sensibilidades (derivadas parciais) obtidos a partir do treinamento de uma rede neural são uma boa alternativa aos métodos estatísticos tradicionais.

Para avaliar o desempenho destes métodos foram treinadas três redes neurais utilizando dados do repositório da UCI *Machine Learning* (FRANK et al., 2011). A partir das redes treinadas os métodos foram executados e as redes neurais foram retreinadas de acordo com o resultado de cada método.

Para problemas de natureza mais simples como o da Iris, por exemplo, todos os métodos apresentaram resultados satisfatórios, identificando claramente as variáveis mais importantes e após o retreinamento das redes o erro médio quadrático obtido foi próximo ao original.

Porém, nos problemas mais complexos com a presença de mais variáveis de entrada como o da CPU Performance e o da Resistência do Concreto os métodos Perturb e Correlação obtiveram os piores resultados no retreinamento e por isso foram considerados os mais ineficientes. Já o algoritmo de Garson apresentou um desempenho satisfatório na determinação da importância das variáveis em todos os problemas. Os métodos de Análise de Sensibilidade e PaD apresentaram resultados praticamente iguais em todos os problemas devido ao fato de utilizarem o mesmo conceito (derivadas parciais) em sua implementação, ambos se destacaram em relação aos demais justamente por identificar com maior precisão tanto as variáveis mais importantes quanto as menos significativas.

Em relação ao desempenho dos métodos apresentados pode-se supor que os métodos Pad e Análise de Sensibilidade obtiveram os melhores resultados devido ao fato de utilizarem mais artifícios para o cálculo da

importância das variáveis ao invés de considerar somente os pesos das conexões como o algoritmo de Garson. Já o método Perturb obteve o pior resultado entre todos os métodos, este resultado pode ser atribuído à sua heurística muito simples de adicionar ruído em uma variável e analisar seu impacto no erro médio quadrático da rede neural.

Os métodos de seleção de características ao identificar a importância das variáveis possibilita a eliminação das menos significativas reduzindo-se o número de entradas da rede. Isto faz com que o treinamento seja mais rápido e ameniza o problema da maldição da dimensionalidade sem que a rede neural perca seu desempenho.

Este trabalho apresentou uma comparação entre quatro métodos de seleção de características utilizando Redes Neurais Artificiais e um estatístico. Existem dezenas de outros métodos de seleção de características utilizando RNAs encontrados na literatura que futuramente podem ser implementados complementando este trabalho.

REFERÊNCIAS

- BARRETO J. M. **Introdução as redes neurais artificiais**, V Escola Regional de Informática da SBC Regional Sul, Santa Maria, Florianópolis, Maringá, p.41-71, 5-10/05/1997.
- BRAGA A.; CARVALHO A.; LUDERMIR T. **Redes Neurais Artificiais: Teoria e Aplicações**. Rio de Janeiro: LTC, 2000, 262p.
- CALOBA, G. M; CALOBA, L. P; SALIBY, E. **Cooperação entre redes neurais artificiais e técnicas 'clássicas' para previsão de demanda de uma série de vendas de cerveja na Austrália**. Rio de Janeiro, v. 22, n. 3, 2002. Disponível em:
<http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382002000300004&lng=en&nrm=iso>. Acesso em: 15 Set. 2011.
- CYBENKO G. Approximations by superpositions of sigmoidal functions. **Mathematics of Control, Signals, and Systems**, v. 2 p 303-314, 1989.
- HAIKIN, S. **Neural Networks: A Comprehensive Foundation**. 2. ed. Mc Master University Hamilton Ontario, Canada: Prentice Hall, 1999. 900p.
- HASSOUN, M. H. **Fundamentals of Artificial Neural Networks**. 1. ed. MIT Press. ,1995. 511p.
- KOVACS K. L. **Redes Neurais Artificiais - Fundamentos e Aplicações**. 4. ed. São Paulo: Livraria da Física, 2002. 174p.
- McCULLOCH W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115-133, 1943.
- MINSK M.; PAPERT S. Perceptrons: an introduction to computational geometry. **MIT Press**, Massachusetts, 1969.
- ROSENBLATT F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychol. Rev.** 1958.
- VALIANT, L. G. A theory of the learnable. **Magazine Communications of the ACM**. v. 27, 1984.
- WIDROW B.; HOFF M.E. Adaptative switching circuits. **Institute of Radio Engineers, Western Electronic Show and Conrention**, v. 4, p. 96-104, 1960.

GARSON, G.D. Interpreting neural network connection weights. **Artificial Intelligence Expert**, v. 6, p 47-51, 1991.

GREVEY, M; DIMOPOULOS, I; LEK, S. (2003). Review and comparison of methods to study the contribution of variables in artificial neural network models. **Ecol. Modell**, v.160, p.249-264.

DIMOPOULOS, Y.; BOURRET, P.; LEK, S. (1995). Use of some sensitivity criteria for choosing networks with good generalization ability. **Neural Processing Letters**, v.2, p.1- 4.

DIMOPOULOS, I.; CHRONOPOULOS, J.; CHRONOPOULOU, A.; LEK, S. (1999). Neural network models to study relationships between lead concentration in grasses and permanent urban descriptors in Athens city (Greece). **Ecological Modelling** v. 120, p.157-165.

FRANK, A; ASSUNCION, A. (2011). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. Disponível em: <http://archive.ics.uci.edu/ml> Acessado em: 10/10/2011.

HRISTEV, R. (2011). ANN Toolbox for Scilab. Disponível em: http://www.scilab.org/contrib/index_contrib.php?page=displayContribution&fileID=166. Acessado em: 20/10/2011.

VALENÇA, M.J.S.; LUDEMIR, T. B. Explicando a relação entre as variáveis de uma rede neural: Iluminando a “Caixa Preta”. **XVII Simpósio Brasileiro de Recursos Hídricos**, São Paulo, Novembro, 2007.

VEELENTRUF, L. P. J. **Analysis and Applications of Artificial Neural Networks**. NJ, USA: Prentice-Hall, 1995. 259 p.

SCILAB CONSORTIUM. **Scilab**. Disponível em: <http://www.scilab.org/>.

YAO, J.; TENG, N.; POH, H.; TAN, C. (1998). Forecasting and analysis of marketing data using neural networks. **Journal of Information Science and Engineering**. v. 14, p. 843-862.

ANEXO A – Algoritmo de Garson

```

function [importancia] = Garson_ConnectionW(w, n)
    importancia = zeros(n(1));
    w = abs(w);
    ctb = zeros(n(1),n(2));//matriz de contribuição
    //armazena matriz de contribuição relativa e soma das contribuições
    relactb = zeros(n(1),n(2)+1);
    p = [1:n(3)];
    tamanho = size(w);

    //Calcula a contribuição de cada entrada em relação a saída

    for i = 1:n(1)
        for j = 1:n(2)
            aux = w(j,i+1,1)*w(1:n(3),j+1,2); //com bias
            //aux = w(j,i,1)*w(1:n(3),j,2); // sem bias
            ctb(i,j) = sum(aux);
        end;
        importanciaCW(i,1) = sum(ctb(i,:));
    end;

    //Calcula a contribuição relativa
    ctb = abs(ctb);
    for i = 1:n(1)
        for j = 1:n(2)
            relactb(i,j) = ctb(i,j)/sum(ctb(:,j));
        end;

        relactb(i,n(2)+1) = (sum(relactb(i,1:n(2))));
    end;

    for i = 1:n(1)
        importancia(i) = (relactb(i,n(2)+1)/sum(relactb(:,n(2)+1)))*100;
    end;
endfunction

```

ANEXO B – Método PaD

```

// Metodo PaD baseado nas derivadas parciais da saída da rede em relação
// à entrada.
// calcula a contribuição SSD (Sum of Square Derivatives) de cada variável
// de entrada.
function ssd = PaD(x, N, W, saida)
//inicializa as variaveis
p = size(x);
d = zeros(p(2),N(1));
ssd = zeros(N(1));

//para cada exemplo de treinamento
for i = 1 : p(2)
//para cada entrada
for k = 1 : N(1)
deriv = 0
for j = 1 : N(2)
//calcula a saída do neurônio escondido
out = saidaNeuronio(x,j,i,W)
//Derivada parcial função sigmoid = y(1-y)
deriv = deriv + (W(1,j+1,2)*(out*(1-out))*W(j,k+1,1))

end

d(i,k) = (saida(1,i)*(1-saida(1,i))) * deriv;

end

end

// calcula a contribuição ssd
for k = 1: N(1)
ssd(k)=0;
for i = 1 :p(2)
ssd(k) = d(i,k)*d(i,k) + ssd(k)

end

end

```

endfunction

ANEXO C – Análise de Sensibilidade

// Metodo Análise de Sensibilidade baseado nas derivadas parciais da variável de saída com relação aos pesos nas conexões

function contri = sensibilidade(**x, y, N, W, saida**)

//inicializa as variaveis

p = size(**x**);
d = zeros(**p**(1),**N**(1));
ssd = zeros(**N**(1));

//para cada exemplo de treinamento

```

for i = 1 : p(2)
    for k = 1 : N(1)
        soma = 0;
        for j = 1 : N(2)
            //calcula a saida do neuronio escondido
            out = saidaNeuronio(x,j,i,W)
            //calcula o erro do neuronio de saida
            e = saida(i) - y(i);
            //Derivada parcial (função sigmoid = y(1-y))
            soma = soma + (((out*(1-out))*W(j,k+1,1)) * ((saida(i)*(1-
saida(i))*(W(1,j+1,2))))*e)
        end
        sen(i,k) = soma;
    end
end

```

//calcula a contribuicao de cada variavel em relacao a saida

```

for k = 1: N(1)
    contri(k)=0;
    for i = 1 :p(2)
        contri(k) = sen(i,k)*sen(i,k) + contri(k)
    end
end

```

endfunction

