Universidade de São Paulo
Escola de Engenharia de São Carlos

Andre Petronilho

Método de Detecção Automática da Quantidade de
Carga em Máquinas de Lavar Roupas

São Carlos
2013

Andre Petronilho

**Método de Detecção Automática da Quantidade de
Carga em Máquinas de Lavar Roupas**

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Ciências, Programa de Engenharia Elétrica

Área de Concentração: Sistemas Dinâmicos
Orientador: Prof. Dr. Marco Henrique Terra

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o Programa de Pós-Graduação de Engenharia Elétrica.

São Carlos
2013

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

P497m Petronilho, Andre
 Método de detecção automática da quantidade de
 carga em máquinas de lavar roupas / Andre Petronilho;
 orientador Marco Henrique Terra. São Carlos, 2013.

 Dissertação (Mestrado) - Programa de Pós-Graduação
 em Engenharia Elétrica e Área de Concentração em
 Sistemas Dinâmicos -- Escola de Engenharia de São
 Carlos da Universidade de São Paulo, 2013.

 1. Redes neurais. 2. Lavadora de roupas. 3.
 Detecção de carga. 4. Sistemas inteligentes. 5.
 Eletrodomésticos. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **ANDRE PETRONILHO**.

Título da dissertação: "Método de detecção automática da quantidade de carga em máquinas de lavar roupas".

Data da defesa: 22/04/2013

Comissão Julgadora:

Resultado:

Prof. Associado **Marco Henrique Terra (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

Aprovado

Prof. Dr. **Daniel Varela Magalhães**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof^a. Associada **Kalinka Regina Lucas Jaquie Castelo Branco**
(Instituto de Ciências Matemáticas e de Computação/ICMC)

Aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica e Presidente da
Comissão de Pós-Graduação:

Prof. Titular **Denis Vinicius Coury**

ESCAR
Serviço de Pós-Graduação
Prof. Titular

Dedicatória

À minha esposa, com amor, admiração e gratidão por sua compreensão, carinho, presença e incansável apoio ao longo do período de elaboração deste trabalho e à minha filha, que apenas com sua presença e sorrisos me inspira e motiva.

Aos meus pais, que nos anos de convivência, muito me ensinaram, contribuindo para meu crescimento pessoal, científico e intelectual.

Agradecimentos

Ao Professor Marco Henrique Terra pela orientação e paciência na realização deste trabalho.

À Escola de Engenharia de São Carlos, pela oportunidade da realização do curso de mestrado.

Aos meus chefes e colegas de trabalho que incentivaram e me ajudaram, disponibilizando tempo e equipamentos para a realização de experimentos.

Resumo

Petronilho, A. Um Método de Estimativa de Quantidade de Carga em uma Máquina de Lavar Roupas. 2013. 78 f . Dissertação de mestrado - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo, 2013.

Nesta dissertação apresentaremos um método para detecção automática da quantidade de carga adicionada em uma lavadora de roupas para que esta possa adequar o seu nível de água e ciclo de lavagem. A máquina na qual este algoritmo foi desenvolvido é uma máquina de eixo vertical (abertura na parte de cima) e utiliza um motor síncrono trapezoidal (BPM do inglês Brushless Permanent Magnet). O algoritmo que será descrito aqui utiliza uma rede neural para inferir a quantidade de carga baseado em informações disponíveis nesta máquina como, corrente do motor, velocidade do cesto e tensão de alimentação, entre outros, essas informações estão disponíveis na maioria dos modelos de máquinas de lavar roupas que utilizam esse tipo de motor. A utilização de um algoritmo para detectar automaticamente e de forma precisa a quantidade de roupas é muito importante, pois dessa forma evita-se o desperdício de insumos e, principalmente, água no processo de lavagem. Além disso apresentaremos os resultados que mostram a diferença entre o uso da rede neural e o método linear chamado planejamento de experimento (DOE do inglês Design of Experiments).

Palavras-Chave: Redes neurais, Lavadora de roupas, Detecção de Carga, Sistemas inteligentes, Eletrodoméstico.

Abstract

Petronilho, A. A Method of Load Estimation in Washing Machines . 2011. 78 f. Master Thesis - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo, 2013.

In this dissertation a method for automatic load size detection will be presented, so the water level and the washing cycles can be chosen by a washing machine. The machine where this algorithm was developed is a top load washing machine that uses a brushless permanent magnet motor (BPM motor). The algorithm that is going to be described here uses a neural network to deduce the load size based on information available on this machine such as, motor current, basket speed, power supply voltage and others. These signals are available on most washing machines that uses this kind of motor. The use of an algorithm that detects automatically and precisely the load amount is very important in order to avoid the waste of soap, bleach and softener and, more importantly, water during the wash task. Moreover the use of the neural network will be compared with a linear methods called DOE (design of experiment). Finally, results showing the difference between both methods are presented.

Keywords: *Neural networks, Washing machine, Load Detection, Intelligent Systems, Home appliances.*

Lista de Figuras

1.1	Máquina de eixo vertical.	2
1.2	Média do consumo de água por ciclo no mundo, em litros.	3
1.3	Média do consumo de eletricidade por ciclo no mundo, em kWh.	4
2.1	Representação de um neurônio real.	8
2.2	Neurônio artificial.	11
2.3	Função degrau.	12
2.4	Função logística.	12
2.5	Representação geométrica da função discriminante.	14
2.6	Adaline.	16
2.7	Perceptron multicamadas.	19
2.8	Configuração de neurônio.	20
2.9	Processo (DOE).	25
2.10	Exemplo de interações entre fatores Moen et al. (1999).	29
2.11	Exemplo de gráfico de efeito.	31
2.12	Exemplo de árvore experimental.	31
2.13	Motor BPM com arco magnético de 180° e corrente de fase de onda quadrada de 120°	34
2.14	Forma de onda das correntes de fase.	34
3.1	Tipos de máquinas de lavar.	39

3.2	Máquina de lavar em corte.	40
3.3	Rotor e estator do motor BPM.	41
3.4	Estator na máquina.	41
3.5	Esquemático do inversor.	41
3.6	Programa leitor de arquivos.	42
3.7	Neural network toolbox.	43
4.1	Momento de inércia de um cilindro.	46
4.2	Comportamento de ω e I entre os passos 3 e 7.	48
4.3	Fluxograma do algoritmo.	49
4.4	Árvore experimental.	52
4.5	Influência dos fatores na corrente.	53
4.6	Influência dos fatores no tempo de desaceleração.	54
4.7	Gráfico pareto do fatores medidos mais relevantes.	55
4.8	Peso real x estimado para fatores mais relevantes.	56
4.9	Gráfico pareto dos 6 fatores medidos mais relevantes.	56
4.10	Peso real x estimado para os 6 fatores mais relevantes.	57
4.11	Resultado do DOE, peso real x estimado.	57
4.12	Resultado do DOE, histograma.	58
4.13	Resultado da rede neural, peso real x estimado sem dados do DOE.	60
4.14	Resultado da rede neural sem dados do DOE, histograma.	60
4.15	Resultado da equação do método DOE com os dados da rede neural, peso real x estimado.	61
4.16	Resultado da rede neural, peso real x estimado com os dados do DOE	61
4.17	Resultado final da rede neural.	63
4.18	Resultado final da rede neural, histograma.	63
4.19	Erro quadratico médio durante treinamento.	64

4.20	Fluxograma do software de treinamento e implementação da rede neural.	65
4.21	Resultado da rede neural, peso real x estimado implementada no MatLab.	66
4.22	Resultado da rede neural implementada no MatLab, histograma.	66
4.23	Características do microcontrolador utilizado.	67

Lista de Tabelas

2.1	Exemplo de matriz experimental.	30
2.2	Exemplo de matriz experimental completa.	32
4.1	Tabela experimental.	51
4.2	Tabela comparativa implementação micro.	68

Sumário

Resumo	v
Abstract	vii
Lista de Figuras	ix
Lista de Tabelas	xiii
1 Introdução	1
1.1 Motivação	1
1.2 Estrutura do Texto	6
1.3 Publicações	6
2 Fundamentos Teóricos	7
2.1 Redes Neurais	7
2.1.1 Perceptron, Adaline e Regra Delta	13
2.1.2 Perceptron Multicamadas	18
2.2 Planejamento de Experimentos	24
2.2.1 Experimentos Com Mais de Um Fator	28
2.2.2 Fatorial Fracionário	32
2.3 Motor Síncrono Trapezoidal	33
2.4 Algoritmos de Estimativa de Carga	36
3 Sistema Utilizado	39
3.1 Máquina de Lavar Roupas	39
3.2 Sistema de Aquisição e Tratamento de Dados	41
4 Resultados	45
4.1 Algoritmo Desenvolvido	45
4.2 Planejamento de Experimentos	50

4.3	Redes Neurais	59
4.4	Avaliação dos Métodos	61
4.5	Métodos Combinados	62
4.6	Implementação da Rede Neural	64
4.6.1	Implementação em MatLab	64
4.6.2	Implementação em C	67
5	Conclusão	71
	Referências Bibliográficas	73
6	Apêndice	75
6.1	Função de Treinamento no Matlab	75
6.2	Função do erro quadratico médio no Matlab	76
6.3	Rede neural implementada no Matlab	77
6.4	Função de multiplicação de matrizes em C	77
6.5	Função de ativação implementada em C	78

Capítulo 1

Introdução

1.1 Motivação

É cada vez maior a preocupação com o meio ambiente, as recentes mudanças climáticas e alguns desastres naturais de grandes proporções, finalmente, chamaram a atenção da população mundial para os impactos das ações humanas sobre o clima do planeta.

Em vários países os governos tentam diminuir os danos causados ao meio ambiente impondo leis que estabelecem um valor máximo de consumo de energia elétrica e água aos produtos eletrônicos e eletrodomésticos vendidos no país, proibindo a venda dos produtos que não atendem essas leis. Outra medida é dar auxílios fiscais aos produtos que consumirem menos do que os limites estipulados, quanto menor o consumo de energia e água, maior o incentivo. No Brasil, não existe a proibição de comercialização de produtos com alto consumo de energia e água, mas existe a classificação segundo um programa de etiquetagem, o selo Procel (Eletrobrás (2009)) de economia de energia, instituído por Decreto Presidencial em 8 de dezembro de 1993. É um produto desenvolvido e concedido pelo Programa Nacional de Conservação de Energia Elétrica, coordenado pelo Ministério de Minas e Energia (MME), com sua Secretaria-Executiva mantida pelas Centrais Elétricas Brasileiras S.A (Eletrobrás). No processo de concessão do Selo Procel, a Eletrobrás conta com a parceria do Instituto Nacional de Metrologia, Normalização e Qualidade Industrial-Inmetro, executor do Programa Brasileiro de Etiquetagem-PBE, cujo principal produto é a Etiqueta Nacional de Conservação de Energia (ENCE), sendo também a Eletrobrás, parceira do Inmetro no desenvolvimento do PBE. Normalmente, os produtos contemplados com o Selo Procel são caracterizados pela faixa “A” da ENCE. O Selo Procel tem por objetivo orientar

o consumidor no ato da compra, indicando os produtos que apresentam os melhores níveis de eficiência energética dentro de cada categoria, proporcionando assim economia na sua conta de energia elétrica. Também estimula a fabricação e a comercialização de produtos mais eficientes, contribuindo para o desenvolvimento tecnológico e a preservação do meio ambiente (Eletrobrás (2009)).

Por isso, as empresas investem muito dinheiro e esforço para encontrar soluções para tornar seus produtos mais eficientes.

Um dos eletrodomésticos que mais consome água e energia é a máquina de lavar roupa, especialmente as máquinas de eixo vertical (abastecimento de roupa por cima) Figura 1.1. Lavar roupa é uma das tarefas domésticas mais difundidas no mundo. Anos atrás, era uma tarefa mecânica árdua, e em algumas regiões do mundo, ainda é. Hoje, as máquinas de lavar fazem este trabalho em muitas casas em todo o mundo com água, eletricidade, substâncias químicas, e tempo de processo como recursos (Pakula e Stamminger (2009)).

Figura 1.1: Máquina de eixo vertical.

Uma das técnicas que contribuem para a diminuição do consumo de água é a detecção automática da quantidade de roupa para que a própria máquina determine o nível de água adequado para cada lavagem, possibilitando um melhor ajuste da quantidade de água utilizada

em cada lavagem, diferentemente do que acontece quando o nível de água é ajustado pelo usuário, que segundo estudos, tende a superestimar a quantidade de água necessária em cada lavagem.

O melhor ajuste do nível de água também contribui na qualidade de lavagem, pois dilui adequadamente os insumos (sabão, alvejante e amaciante), além disso também ajuda na diminuição do consumo de energia em máquinas que utilizam água aquecida, seja água que foi aquecida em um *boiler* e fornecida para a máquina (sistema mais comum nos Estados Unidos e Europa) ou em máquinas que utilizam uma resistência elétrica para aquecer a água (sistema mais comum no Brasil), porque a energia necessária para se aquecer uma menor quantidade de água também é menor. Outra vantagem que a detecção automática de água traz é a maior facilidade de interação do usuário com a máquina, diminuindo as etapas necessárias para se configurar a máquina adequadamente.

Torna-se óbvio que o desenvolvimento de um algoritmo de detecção de quantidade de carga preciso é muito importante para este tipo de tarefa. Por exemplo, com um erro de 1 cm na quantidade de água, normalmente a quantidade de água é medido pela altura da coluna de água dentro do cesto da máquina, em uma máquina com capacidade de 147,25 litros (máquina de grande capacidade de carga), representa cerca de 2,7 litros de água cada vez que a máquina é enchida. Isso acontece 3 ou mais vezes para cada ciclo de lavagem. Considerando-se a estimativa que existam bem mais de 600.000.000 (Pakula e Stamminger (2009)) de máquinas de lavar em todo o mundo, isso pode fazer uma diferença muito significativa no consumo de água mundial. A média do consumo de água e de energia por ciclo de lavagem para máquina de lavar roupa ao redor do mundo pode ser visto nas Figuras 1.2 e 1.3 (Pakula e Stamminger (2009)).

Figura 1.2: Média do consumo de água por ciclo no mundo, em litros.

Figura 1.3: Média do consumo de eletricidade por ciclo no mundo, em kWh.

No Brasil a média do consumo de água por ciclo é de 138 litros e o consumo de eletricidade é de 0,41KWh, segundo dados de 2011 do Inmetro.

Dois métodos em especial chamam a atenção quando se pensa no desenvolvimento de algoritmos, o primeiro é um método largamente utilizado na indústria e conhecido como planejamento de experimentos, o outro são as redes neurais artificiais que são bastante estudadas no meio acadêmico, porém não são tão popular na indústria, apesar de existirem algumas aplicações em produção com esse tipo de tecnologia.

Em qualquer área de pesquisa, está-se sempre interessado em saber quais variáveis são importantes em algum estudo que se esteja realizando, assim como limites inferior e superior de valores dessas variáveis. Por exemplo, em um processo de cura de resinas termorrígidas, a temperatura de cura e a quantidade de catalisador são variáveis importantes na determinação do tempo total de cura da resina polimérica. É necessário escolher, de acordo com a conveniência do pesquisador, limites inferior e superior de valores da temperatura de cura e da quantidade de catalisador. Corridas experimentais (rodadas de testes) devem ser realizadas afim de se obter dados do tempo de cura para cada combinação dos limites de valores da temperatura de cura e da quantidade de catalisador. A partir desses resultados, técnicas estatísticas devem ser utilizadas de modo a se concluir algo em relação à dependência do tempo de cura com as variáveis analisadas. O planejamento experimental é uma dessas técnicas, que atualmente vem sendo usada em grande escala. Através dele, pesquisadores podem determinar as variáveis que exercem maior influência no desempenho de um determinado processo, tendo como resultado (Calado e Montgomery (2003)):

1. redução da variação do processo e melhor concordância entre os valores nominais obtidos e os valores pretendidos;
2. redução do tempo do processo;
3. redução do custo operacional;
4. melhoria no rendimento do processo.

Algumas aplicações típicas do planejamento de experimentos são:

1. avaliação e comparação de configurações básicas de projeto;
2. avaliação de diferentes materiais;
3. seleção de parâmetros de projeto;
4. determinação de parâmetros de projeto que melhorem o desempenho de produtos.

No desenvolvimento de produtos, especialmente produtos mais complexos, com alta tecnologia embarcada, muitas vezes as informações sobre o produto ou os processos necessitam ser obtidas empiricamente, ou seja baseadas na experiência. Para realizar esse trabalho é necessário projetar experimentos, coletar dados e analisá-los.

Uma técnica utilizada para planejar experimentos, que tenta contemplar os objetivos citados acima, é o Planejamento de Experimentos (em inglês *Design of Experiments*, DOE). Esta técnica é utilizada para definir quais dados, em que quantidade e em que condições devem ser coletados durante um determinado experimento. Portanto, é uma técnica de fundamental importância para a indústria, pois seu uso permite resultados mais confiáveis economizando dinheiro e tempo, parâmetros fundamentais em tempos de grande concorrência. A sua aplicação no desenvolvimento de novos produtos é muito importante, onde uma maior qualidade dos resultados dos testes pode levar a um projeto com desempenho superior seja em termos de suas características funcionais como também sua robustez.

O segundo método citado, as redes neurais artificiais, é extremamente versátil, podendo ser utilizado para resolver uma grande variedade de problemas, por serem modelos adaptativos treináveis, assim como o método anterior, utiliza informações obtidas empiricamente e além disso consegue inferir soluções que não foram apresentadas no processo de treinamento, tornando esse

método muito robusto, algo muito buscado na indústria, principalmente em produtos de alto volume de produção que podem apresentar variações devido ao processo de produção.

Uma grande parte dos processos e sistemas utilizados atualmente, por serem cada vez mais complexos, não são lineares, o que torna o uso de uma rede neural extremamente atraente, devido à sua natureza não linear. O fato da rede neural possuir grande paralelismo, faz com que o processamento seja rápido mesmo com sistemas de grande complexidade.

Neste trabalho, primeiramente o desenvolvimento do algoritmo de estimativa de carga será apresentado, onde durante a execução deste algoritmo parâmetros relevantes da máquina, como corrente, tensão e temperatura, são medidos, em seguida será mostrado o planejamento de experimentos para a determinação de um modelo linear para a estimativa de quantidade de carga tendo como entrada os parâmetros medidos durante a execução do algoritmo, após isso uma rede neural será utilizada para estimar a quantidade de carga, finalmente os dois métodos são comparados e a rede neural resultante é implementada em um microcontrolador.

1.2 Estrutura do Texto

Este texto está organizado da seguinte maneira: No Capítulo 1 é exposta a motivação deste trabalho, no Capítulo 2 é apresentada uma revisão bibliográfica dos principais assuntos abordados nesta pesquisa: redes neurais, planejamento de experimentos (DOE), modelagem do motor e o algoritmo de detecção de quantidade de carga. Já no Capítulo 3 é descrito o sistema utilizado, no Capítulo 4 o algoritmo desenvolvido é descrito e os resultados são apresentados, por fim no Capítulo 5 são apresentadas as conclusões e um apêndice com o código fonte da implementação das redes neurais.

1.3 Publicações

Congresso Internacional

Petronilho, A. e Terra, M. H. (2010). Automatic control applied to home appliances. 2010 IEEE Multi-Conference on System and Control, Yokohama (Petronilho e Terra (2010)).

Patente

Petronilho, A. e Dias, M. A. (2009). PI0905317-4 - Método de estimação de quantidade de carga em máquina de lavar roupas (Petronilho e Dias (2009)).

Capítulo 2

Fundamentos Teóricos

2.1 Redes Neurais

O cérebro humano é considerado o mais fascinante processador existente, sendo composto por aproximadamente 10 bilhões de neurônios. Todas as funções e movimentos do organismo estão relacionados ao funcionamento destas pequenas células. Os neurônios estão conectados uns aos outros através de sinapses, e juntos formam uma grande rede, chamada REDE NEURAL. As sinapses transmitem estímulos através de diferentes concentrações de Na^+ (Sódio) e K^+ (Potássio), e o resultado disto pode ser estendido por todo o corpo humano. Esta grande rede proporciona uma fabulosa capacidade de processamento e armazenamento de informação.

Os principais componentes dos neurônios são (Figura 2.1): os dendritos, que são constituídos por vários finos prolongamentos que formam a árvore dendrital. A principal função dos dendritos consiste de captar, de forma contínua, os estímulos vindos de diversos outros neurônios ou do próprio meio externo onde os mesmos podem estar em contato (neurônios sensitivos). O corpo celular, que é incumbido de processar todas as informações advindas dos dendritos a fim de produzir um potencial de ativação que indicará se o neurônio poderá disparar um impulso elétrico ao longo de seu axônio. E o axônio, que é constituído por um único prolongamento, cuja missão é conduzir os impulsos elétricos para outros neurônios conectores ou para aqueles que se conectam diretamente com o tecido muscular (neurônios efetadores). A sua terminação é também constituída de ramificações denominadas terminações sinápticas. (Silva et al. (2010))

As sinapses se configuram como as conexões que viabilizam a transferência de impulsos elétricos do axônio de um neurônio para os dendritos de outros. Deve-se mencionar que inexiste

Figura 2.1: Representação de um neurônio real.

contato físico entre os neurônios na junção sináptica, sendo que elementos neurotransmissores liberados são os responsáveis por ponderar a transmissão de impulsos elétricos de um neurônio para o outro. De fato, a funcionalidade de um neurônio fica a mercê do comportamento dessas ponderações sinápticas que são dinâmicas e dependentes da química cerebral (Hodgkin e Huxley (1952)).

No momento em que a célula nervosa é estimulada (despolarizada), além de um limiar de ativação (-55 mV), que é ocasionado pela variação de concentração interna de íons de sódio (Na^+) e potássio (K^+), há então o disparo de um impulso elétrico que será propagado ao longo de todo o seu axônio, cuja amplitude máxima alcançará o valor de 35 mV (Kandel et al. (2000)).

A estrutura das redes neurais artificiais procura aproximar o processamento dos computadores a modelos conhecidos de sistemas nervosos biológicos e do próprio cérebro humano. Uma rede neural artificial é composta por várias unidades de processamento, cujo funcionamento é bastante simples. Essas unidades geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões, o comportamento inteligente de uma Rede Neural Artificial vem das interações entre as unidades de processamento da rede.

As características mais relevantes das redes neurais artificiais, que motivam sua utilização são descritas abaixo:

- modelos adaptativos treináveis: seus peso sinápticos são ajustados com a apresentação de dados de treinamento relacionados ao comportamento do processo.

- são capazes de generalização diante de informações incompletas: consegue estimar soluções que não foram apresentadas durante o processo de treinamento, generalizando o conhecimento adquirido.
- podem representar domínios complexos (não lineares): redes de múltiplas camadas conseguem modelar sistemas não lineares, algo muito difícil de se fazer com métodos tradicionais.
- robustas: mesmo com parte de sua estrutura sendo corrompida consegue continuar tendo resultados corretos devido ao grande número de interconexões entre seus neurônios artificiais.
- organização de dados: consegue agrupar padrões que apresentam particularidades em comum.
- são capazes de fazer armazenamento associativo de informações.
- facilidade de prototipagem: pode ser prototipada tanto em hardware quanto em software, pois, após o processo de treinamento, os seus resultados são normalmente obtidos por algumas operações matemáticas elementares.
- possuem grande paralelismo: o que lhe conferem rapidez de processamento.

Pode-se citar como as principais aplicações das redes neurais artificiais:

- a) Aproximador universal de funções:** o objetivo consiste em mapear o relacionamento funcional entre as variáveis (tipicamente reais) de um sistema a partir de um conjunto conhecido de seus valores representativos. As aplicações são as mais diversas possíveis, sendo que envolvem normalmente o mapeamento de processos, cuja modelagem por técnicas convencionais são de difícil obtenção;
- b) Controle de processos:** O objetivo consiste em identificar ações de controle que permitam o alcance dos requisitos de qualidade, eficiência e segurança do processo. Entre as várias aplicações disponíveis destacam-se os controles empregados em robótica, aeronaves, elevadores, eletrodomésticos, satélites, etc;
- c) Reconhecimento / classificação de padrões:** O objetivo deste tipo de aplicação consiste de associar um padrão de entrada (amostra) para uma das classes previamente definidas, como acontece em reconhecimento de imagens, voz, escrita, etc. Neste caso, o problema a ser tratado possui um conjunto discreto e conhecido das possíveis saídas desejadas;

- d) Agrupamento de dados (clusterização): O objetivo nesta circunstância consiste da identificação e detecção de similaridades e particularidades entre os diversos padrões de entrada a fim de possibilitar seu agrupamento. Como exemplo, pode-se citar os problemas que envolvem identificação automática de classes e garimpagem de dados;
- e) Sistemas de previsão: O objetivo consiste em estimar valores futuros de um processo levando-se em consideração diversas medidas previas observadas em seu domínio. Entre as aplicações disponíveis enquadram-se a previsão de series temporais, previsões de mercados financeiros, previsões climáticas, etc;
- f) Otimização de sistemas: O alvo consiste em minimizar ou maximizar uma função custo (objetivo) obedecendo também eventuais restrições que são impostas para o correto mapeamento do problema. Entre as classes de otimização que podem ser tratadas por redes neurais artificiais destacam-se problemas de otimização restrita, programação dinâmica e otimização combinatorial;
- g) Memórias associativas: O objetivo consiste em recuperar padrões corretos mesmo se os seus elementos constituintes forem apresentados de forma incerta ou imprecisa. Entre as aplicações citam-se aquelas relacionadas ao processamento de imagens, a transmissão de sinais, identificação de caracteres manuscritos, etc. (Silva et al. (2010)).

O modelo de neurônio mais utilizado nas diferentes arquiteturas de redes neurais artificiais é o modelo proposto por McCulloch e Pitts (1943), uma representação deste modelo pode ser visto na Figura 2.2.

A operação desta unidade de processamento, pode ser resumida da seguinte maneira:

Os sinais de entrada $X_1, X_2, \dots, X_n$, que são sinais ou medidas provenientes do meio externo, são apresentados ao neurônio e para melhorar a eficiência computacional são geralmente normalizados. Cada sinal é multiplicado por um número, ou pesos sinápticos $W_1, W_2, \dots, W_n$, que indica a sua influência na saída da unidade, então é feita a soma ponderada dos sinais que produz um nível de atividade. Se este nível de atividade exceder um certo limite (*threshold*) θ a unidade produz um potencial excitatório se $u \geq \theta$, caso contrário, o potencial será inibitório. Ao sinal u é aplicada uma função de ativação $g(\cdot)$, cujo objetivo é limitar a saída do neurônio dentro do intervalo de valores razoáveis a serem assumidos, gerando um sinal de saída y . O resultado produzido pelos neurônios é similar à solução das seguintes equações:

Figura 2.2: Neurônio artificial.

$$u = \sum_{i=1}^n w_i \cdot x_i - \theta \quad (2.1)$$

$$y = g(u), \quad (2.2)$$

sendo que a função de ativação pode ser dividida em dois grupos principais: funções diferenciáveis por partes e diferenciáveis. No primeiro grupo a função com maior destaque é a função degrau Figura 2.3, já no segundo grupo uma das funções mais usada é a logística Figura 2.4, descritas abaixo:

$$g(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{se } u < 0 \end{cases} \quad (2.3)$$

$$g(u) = \frac{1}{(1 + e^{-\beta u})}, \quad (2.4)$$

sendo que β influencia na inclinação da função logística.

Arquiteturas neurais são tipicamente organizadas em camadas, as quais são nomeadas da seguinte forma:

- Camada de Entrada: onde os padrões são apresentados à rede.

Figura 2.3: Função degrau.

Figura 2.4: Função logística.

- Camadas Intermediárias ou Escondidas: onde é feita a maior parte do processamento, através das conexões ponderadas. Essas camadas extraem as características do processo ou sistema a ser inferido.
- Camada de Saída: onde o resultado final é concluído e apresentado.

Dependendo da disposição de seus neurônios e das suas formas de interligação, as redes neurais podem ser divididas em algumas arquiteturas principais: **redes *feedforward* de camada simples**, com uma camada de entrada e uma camada de neurônios e que tem como principais aplicações a classificação de padrões e a filtragem linear, pode-se citar o *Perceptron* e o *Adaline* como as principais representantes deste tipo de rede neural; **redes *feedforward* de camadas múltiplas** que são caracterizadas pela presença de pelo menos uma camada escondida de neurônios, com uma gama grande de aplicações, como, aproximação de funções, classificação de padrões, identificação de sistemas, otimização, robótica e controle de processos. Essa rede é uma das arquiteturas mais utilizada, sendo que o *Perceptron* multicamadas é o tipo mais famoso dentre as redes desta arquitetura, além das redes de base radial (RBF) que também são bastante utilizadas nessa arquitetura; **redes recorrentes ou realimentadas**, nesta arquitetura as saídas dos neurônios são realimentadas como sinais de entrada para outros neurônios, assim essas redes são ideais para aplicações em previsão de séries temporais, otimização e identificação de sistemas e em controle de processos, onde a rede de Hopfield e *Perceptron* multicamadas com realimentação estão entre os principais tipos de redes; e **redes reticuladas**, nesta arquitetura a disposição dos neurônios está diretamente relacionada com o processo de ajuste dos pesos e limiares e tem aplicações em problemas de agrupamento (*clustering*), reconhecimento de padrões, otimização de sistemas, e grafos.

A propriedade mais importante das redes neurais é a habilidade de aprender com seu ambiente e com isso melhorar seu desempenho. Isso é feito através de um processo iterativo de

ajustes aplicado a seus pesos, o treinamento. O aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma classe de problemas.

Denomina-se algoritmo de aprendizado um conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados.

A rede neural se baseia nos dados para extrair um modelo geral. Portanto, a fase de aprendizado deve ser rigorosa e verdadeira, a fim de se evitar modelos espúrios. Todo o conhecimento de uma rede neural está armazenado nas sinapses, ou seja, nos pesos atribuídos às conexões entre os neurônios. De 60 a 90% do total de dados deve ser separado para o treinamento da rede neural, dados estes escolhidos aleatoriamente, a fim de que a rede aprenda as regras e não decore exemplos. O restante dos dados só é apresentado à rede neural na fase de testes a fim de que ela possa deduzir corretamente o inter-relacionamento entre os dados.

Os processos de treinamento mais conhecidos são:

- Treinamento supervisionado: quando é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada.
- Treinamento não supervisionado: (auto-organização), quando não existe um agente externo indicando a resposta desejada para os padrões de entrada.
- Treinamento com reforço: quando um crítico externo avalia a resposta fornecida pela rede.
- Aprendizagem usando lote de padrões (*off-line*): quando o ajuste dos pesos da rede só são realizados após a apresentação de todo o conjunto de treinamento.
- Aprendizagem usando padrão-por-padrão (*on-line*): os ajustes dos pesos e limiares das redes são efetuados após a apresentação de cada amostra de treinamento.

2.1.1 Perceptron, Adaline e Regra Delta

O primeiro modelo para aprendizagem supervisionada de redes neurais, o *Perceptron*, foi proposto por Rosenblatt em 1958, ele é a forma mais simples de uma rede neural usada para classificação de padrões linearmente separáveis. A sua simplicidade é devida a ser constituída de apenas um neurônio com pesos sinápticos ajustáveis e uma polarização. O *Perceptron*, de

um único neurônio é limitado a desempenhar classificação de padrões com apenas duas classes, define-se classes C_1 e C_2 , usando, por exemplo, a função de ativação degrau bipolar o *Perceptron* pode atribuir o valor -1 para representar as amostras pertencentes à classe C_1 , e atribuir 1 para as amostras pertencentes a classe C_2 , isso é feito pela rede ponderando as entradas $\{x_i\}$, que representam informações sobre o comportamento do processo a ser mapeado, pelos pesos sinápticos $\{w_i\}$, depois é adicionado o limiar de ativação $\{\theta\}$ e o resultado é repassado como argumento da função de ativação.

A separação dessas duas classes neste caso é uma reta, dada pela equação:

$$w_1x_1 + w_2x_2 - \theta = 0. \quad (2.5)$$

A Equação (2.5) pode ser escrita da seguinte forma.

$$x_2 = -\frac{w_1}{w_2}x_1 + \frac{\theta}{w_2}. \quad (2.6)$$

A representação geométrica desta separação linear é mostrada na Figura 2.5, onde pode-se notar que os pesos determinam a inclinação da reta e o limiar de ativação determina o “*offset*” da reta.

Figura 2.5: Representação geométrica da função discriminante.

Para possibilitar a classificação de mais do que duas classes pode-se utilizar a expansão da camada computacional de saída do *Perceptron* para incluir mais do que um neurônio, entretanto, as classes têm que ser linearmente separáveis para se ter um desempenho adequado, pois aumentando o número de entradas obtem-se hiperplanos com fronteiras de separação no lugar

da reta.

A regra de aprendizado de Hebb é utilizada para fazer o treinamento do *Perceptron* ajustando seus pesos e limiar.

Para cada amostra de treinamento apresentado para a rede é calculado o valor de $\Delta w_i(t)$ e $\Delta \theta(t)$ e ajustados os pesos e o limiar, conforme as equações 2.7 e 2.8 abaixo. Caso a saída produzida pelo *Perceptron* for coincidente com a saída desejada, tem-se um ajuste excitatório proporcional aos valores de seus sinais de entrada, caso contrário tem-se um ajuste inibitório.

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \quad (2.7)$$

$$\theta(t+1) = \theta(t) + \Delta \theta(t) \quad (2.8)$$

sendo que os valores de Δ são calculados da seguinte forma:

$$\Delta w_i(t) = \Delta \theta(t) = \eta(d^{(k)} - y)\mathbf{x}^{(k)} \quad (2.9)$$

sendo $\mathbf{x}^{(k)} = [-1 \ x_1^{(k)} \ x_2^{(k)} \dots \ x_n^{(k)}]^T$ a k-ésima amostra de treinamento;

$d^{(k)}$ é o valor desejado para a k-ésima amostra de treinamento;

y é o valor de saída produzida pelo *Perceptron*;

η é uma constante que define a taxa de aprendizagem da rede, ou seja o quão rápido o processo de treinamento da rede estará sendo conduzido no sentido da convergência.

O processo de treinamento ajusta continuamente o hiperplano de separação até que este esteja posicionado de tal forma a permitir dividir as duas classes.

Uma generalização importante do algoritmo de treinamento do *Perceptron* foi apresentado por Widrow e Hoff (1960) como o processo de aprendizagem “*least mean square*” (LMS), também conhecido como regra delta. A diferença funcional principal com a regra de aprendizagem do *Perceptron* é a forma como a saída do sistema é utilizado na regra de aprendizagem. A regra de aprendizado do *Perceptron* usa a saída da função de limiar (quer seja -1 ou 1) para a aprendizagem. A regra delta utiliza a saída antes do seu mapeamento para os valores de saída -1 ou 1. A regra de aprendizado foi aplicada ao “*adaptive linear element*”, também chamado Adaline, o qual foi desenvolvida por Widrow e Hoff. Em uma implementação física simples

(Figura 2.6) este dispositivo consistia de um conjunto de resistências controláveis conectadas a um circuito que pode somar as correntes provocadas pelos sinais de tensão de entrada. O somador é seguido de um “quantizador” que tem como saída -1 ou 1, dependendo da polaridade da soma.

Figura 2.6: Adaline.

O objetivo deste dispositivo é o de produzir um determinado valor $y = d^k$ em sua saída quando o conjunto de valores $x_i^k, i = 1, 2, \dots, n$, é aplicado nas entradas. O problema é determinar os coeficientes $w_i = 0, 1, \dots, n$, de tal forma que a resposta seja correta para um grande número de conjuntos de sinais escolhidos arbitrariamente. Se um mapeamento exato não é possível, o erro médio deve ser minimizado, por exemplo, no sentido dos mínimos quadrados. Uma operação adaptativa significa que existe um mecanismo pelo qual o w_i pode ser ajustado, normalmente de forma iterativa, para atingir os valores corretos. Para a Adaline, Widrow introduziu a regra delta para ajustar os pesos.

Para uma rede de camada única, com uma unidade de saída com uma função de ativação linear, a saída é simplesmente dada por

$$y = \sum_{i=1}^n w_i \cdot x_i + \theta. \quad (2.10)$$

Essa rede simples é capaz de representar uma relação linear entre o valor da unidade de saída e o valor das unidades de entrada. Aplicando o limiar no valor de saída, um classificador pode ser construído (como a Adaline de Widrow), mas aqui a relação linear e o uso da rede para uma tarefa de aproximação de função será o foco principal. Nos espaços de entrada de várias

dimensões a rede representa um hiperplano e ficará claro que unidades de várias saídas também podem ser definidas.

Suponha que quer-se treinar a rede de tal forma que o hiperplano de separação fique posicionado no melhor lugar possível para um conjunto de amostras de treinamento que consistem de valores de entrada x^k e saídas desejadas d^k . Para cada amostra de entrada dada, a saída da rede difere do valor alvo por $(d^k - y^k)$, onde y^k é a saída atual para este padrão. A regra delta agora usa um custo ou função de erro com base nessas diferenças para ajustar os pesos. A função de erro, como indica o nome "*least mean square (LMS)*", é a soma do erro quadrático. Ou seja, o erro total E é definido como sendo:

$$E = \sum_k E^k = \frac{1}{2} \sum_k (d^k - y^k)^2, \quad (2.11)$$

sendo que o índice k varia dentro do conjunto de padrão de entrada e E^k representa o erro no padrão k . O procedimento LMS encontra o valor de todos pesos que minimizam a função erro por um método chamado gradiente descendente. A ideia é fazer uma mudança no peso proporcional ao negativo da derivada do erro medido no padrão atual respectivo a cada peso:

$$\Delta_k w_i = -\gamma \frac{\partial E^k}{\partial w_i} \quad (2.12)$$

sendo γ a constante de proporcionalidade. A derivada é

$$\frac{\partial E^k}{\partial w_i} = \frac{\partial E^k}{\partial y^k} \frac{\partial y^k}{\partial w_i}. \quad (2.13)$$

Segue da Equação (2.10) que,

$$\frac{\partial y^k}{\partial w_i} = x_i \quad (2.14)$$

e segue de (2.11) que

$$\frac{\partial E^k}{\partial y^k} = -(d^k - y^k) \quad (2.15)$$

e portanto pode-se escrever (2.12) como

$$\Delta_k w_i = \gamma \delta^k x_i \quad (2.16)$$

sendo $\delta^k = d^k - y^k$ a diferença entre a saída desejada e a saída atual do padrão k .

A regra delta modifica os pesos adequadamente para as saídas atuais, com base na saída desejada independentemente de sua polaridade e para ambas as unidades de entrada e saída sendo elas contínuas e binárias. Estas características possibilitaram uma grande variedade de novas aplicações.

2.1.2 Perceptron Multicamadas

Uma rede de camada única tem severas restrições quanto à classe de tarefas que podem ser executadas, ao contrário o *Perceptron* multicamadas é uma das arquiteturas de rede neural mais versátil, a Figura 2.7 mostra uma representação genérica deste tipo de rede, podendo ser utilizada em diversos tipos de problemas, como:

1. Aproximação de funções,
2. Reconhecimento de padrões,
3. Identificação e controle de processos,
4. Previsão de séries temporais,
5. Otimização de sistemas utilizados.

Como a aproximação de funções é o tipo de problema de maior interesse para este trabalho, este capítulo concentra-se em redes *feedforward* com múltiplas camadas de unidades de processamento e no seu processo de treinamento, o *back-propagation*, cuja idéia central por trás desta solução é que os erros para as unidades da camada escondida são determinados propagando-se os erros das unidades da camada de saída no sentido da camada de entrada. Por esta razão, o método é geralmente chamado de *back-propagation*. E também pode ser considerado como uma generalização da regra delta para as funções de ativação não-linear e redes multicamadas.

Uma rede do tipo *Perceptron* multicamadas tem três características distintas:

1. O modelo de cada neurônio na rede tem uma não-linearidade na saída. Um ponto importante para ser enfatizado é que neste caso a não-linearidade é diferenciável, diferentemente

da função de ativação degrau utilizada no *Perceptron* de Rosenblatt. A presença da não-linearidade é importante porque caso contrário a relação entrada-saída da rede poderia ser reduzida a aquela de uma rede de uma camada.

2. A rede contém uma ou mais camadas escondidas de neurônios, que não são parte da entrada ou da saída da rede. Esses neurônios escondidos possibilitam que a rede aprenda tarefas complexas extraindo progressivamente informações mais relevantes dos padrões de entrada.
3. A rede exibe um alto grau de conectividade, determinada pelas sinapses da rede. Uma mudança na conectividade da rede requer uma mudança na população das conexões sinápticas ou em seus pesos.

Para a dedução da regra delta generalizada utiliza-se a notação vista na Figura 2.7,

Figura 2.7: Perceptron multicamadas.

sendo j o neurônio de cada uma das L camadas e pode ser representado como na Figura 2.8

Figura 2.8: Configuração de neurônio.

e $g(\cdot)$ representa uma função de ativação que deve ser contínua e diferenciável em todo seu domínio, por exemplo a função logística ou tangente hiperbólica.

O sinal de erro na saída do neurônio j na iteração k (apresentação do k -ésimo padrão de treinamento) é definido por

$$e_j(k) = d_j(k) - y_j(k), \text{ neurônio } j \text{ é uma saída.} \quad (2.17)$$

Assim a soma dos erros quadráticos da rede é escrita da seguinte forma

$$E(k) = \frac{1}{2} \sum_{j \in C} e_j^2(k) \quad (2.18)$$

sendo que o conjunto C inclui todos os neurônios na camada de saída da rede. Seja K o número total de padrões de treinamento, o erro quadrático médio é calculado como

$$E_{av} = \frac{1}{K} \sum_{k=1}^K E(k). \quad (2.19)$$

O objetivo do processo de treinamento é o de ajustar os parâmetros livres da rede neural minimizando E_{av} . Considera-se um método simples para tal, ajustando os pesos a cada padrão de treinamento apresentado.

Considere $v_j(k)$ o nível de ativação interna do neurônio, produzido na entrada da função de ativação não-linear do neurônio j como sendo

$$v_j(k) = \sum_{i=0}^p w_{ji}(k) y_i(k) \quad (2.20)$$

com p sendo o total de entradas (excluindo o *threshold*) aplicadas ao neurônio j . O peso sináptico w_{j0} (correspondendo à entrada fixa $y_0 = -1$) é igual ao threshold θ_j aplicado ao neurônio j . Portanto o sinal $y_j(k)$ na saída do neurônio j na iteração k é

$$y_j(k) = g(v_j(k)). \quad (2.21)$$

De maneira similar ao algoritmo LMS o algoritmo *back-propagation* aplica uma correção $\Delta w_{ij}(k)$ no peso sináptico $w_{ij}(k)$, que é proporcional ao gradiente instantâneo $\partial E(k)/\partial w_{ij}(k)$. De acordo com a regra da cadeia, pode-se expressar esse gradiente da seguinte maneira

$$\frac{\partial E(k)}{\partial w_{ji}(k)} = \frac{\partial E(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial w_{ji}(k)}. \quad (2.22)$$

O gradiente $\partial E(k)/\partial w_{ij}(k)$ representa um fator de sensibilidade, determinando a direção de procura no espaço dos pesos para o peso sináptico w_{ij} .

Derivando os dois lados da Equação (2.18) em relação a $e_j(k)$, obtém-se

$$\frac{\partial E(k)}{\partial e_j(k)} = e_j(k). \quad (2.23)$$

Derivando os dois lados da Equação (2.17) em relação a $y_j(k)$, obtém-se

$$\frac{\partial e_j(k)}{\partial y_j(k)} = -1. \quad (2.24)$$

Continuando, derivando a Equação (2.21) em relação a $v_j(k)$, encontra-se

$$\frac{\partial y_j(k)}{\partial v_j(k)} = g'_j(v_j(k)). \quad (2.25)$$

Finalmente, derivando Equação (2.20) em relação a $w_{ji}(k)$, encontra-se

$$\frac{\partial v_j(k)}{\partial w_{ji}(k)} = y_i(k). \quad (2.26)$$

Portanto, o uso das equações 2.23 até 2.26 em 2.22, gera

$$\frac{\partial E(k)}{\partial w_{ji}(k)} = -e_j(k)g'_j(v_j(k))y_i(k). \quad (2.27)$$

A correção $\Delta w_{ji}(k)$ aplicada à $w_{ji}(k)$ é definida pela regra delta

$$\Delta w_{ji}(k) = -\eta \frac{\partial E(k)}{\partial w_{ji}(k)} \quad (2.28)$$

sendo η uma constante que determina a taxa de aprendizagem. O uso do sinal de menos na Equação (2.28) indica que o gradiente decresce no espaço dos pesos. Consequentemente, o uso da equação 2.27 na 2.28 gera

$$\Delta w_{ji}(k) = \eta \delta_j(k) y_i(k) \quad (2.29)$$

sendo $\delta_j(k)$ o gradiente local definido por

$$\delta_j(k) = -\frac{\partial E(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} = e_j(k)g'_j(v_j(k)). \quad (2.30)$$

Das equações 2.29 e 2.30 nota-se que um fator importante envolvido no cálculo da correção do peso $\Delta w_{ji}(k)$ é o sinal de erro $e_j(k)$ na saída do neurônio j . Neste contexto pode-se identificar dois casos distintos, dependendo onde o neurônio j está localizado na rede. No primeiro caso, o neurônio j é um nó de saída, esse caso é simples de lidar, pois cada nó de saída da rede é fornecido com uma saída desejada, fazendo com que seja direto o cálculo do erro associado à saída do neurônio. No segundo caso, o neurônio j está na camada escondida. Mesmo que neurônios escondidos não sejam diretamente acessíveis, eles compartilham a responsabilidade em qualquer erro na saída da rede (Haykin (1994)).

Caso I: Neurônio j na camada de saída

Quando o neurônio j está localizado na camada de saída da rede, ele é fornecido com uma saída desejada, portanto pode-se usar a equação 2.17 para calcular o sinal de erro $e_j(k)$. Tendo determinado o erro $e_j(k)$ é direto o cálculo do gradiente local $\delta_j(k)$ usando a Equação (2.30) (Haykin (1994)).

Caso II: Neurônio j na camada escondida

Quando o neurônio j está localizado na camada escondida da rede, não existe uma resposta desejada para este neurônio, conseqüentemente, o sinal de erro teria de ser determinado recursivamente em termos dos sinais de erro de todos os neurônios nos quais aquele neurônio escondido está diretamente conectado. De acordo com a Equação (2.30), pode-se redefinir o gradiente local $\delta_j(k)$ para o neurônio escondido j como

$$\delta_j(k) = -\frac{\partial E(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} = -\frac{\partial E(k)}{\partial y_j(k)} g'_j(v_j(k)), \text{ neurônio } j \text{ é um nó escondido} \quad (2.31)$$

sendo que na segunda igualdade utiliza-se a Equação (2.25). Para calcular a derivada parcial $\partial E(k)/\partial y_j(k)$, pode-se prosseguir da seguinte forma:

Para diferenciar um neurônio escondido de um outro na camada de saída, irá ser utilizado os índices j e r respectivamente. Reescrevendo a Equação (2.18) obtém-se

$$E(k) = \frac{1}{2} \sum_{r \in C} e_r^2(k), \text{ neurônio } r \text{ é um nó de saída.} \quad (2.32)$$

Derivando a Equação (2.32) em relação a $y_j(k)$, obtém-se

$$\frac{\partial E(k)}{\partial y_j(k)} = \sum_r e_r(k) \frac{\partial e_r(k)}{\partial y_j(k)}. \quad (2.33)$$

Utilizando a regra da cadeia novamente, agora para a derivada parcial $\partial e_r(k)/\partial y_j(k)$ pode-se reescrever 2.33 da seguinte forma

$$\frac{\partial E(k)}{\partial y_j(k)} = \sum_r e_r(k) \frac{\partial e_r(k)}{\partial v_r(k)} \frac{\partial v_r(k)}{\partial y_j(k)}. \quad (2.34)$$

Como o neurônio r é um nó de saída pode-se escrever o erro de forma direta

$$e_r(k) = d_r(k) - y_r(k) = d_r(k) - g_r(v_r(k)), \text{ neurônio } r \text{ é uma saída.} \quad (2.35)$$

Portanto,

$$\frac{\partial e_r(k)}{\partial v_r(k)} = -g'_r(v_r(k)). \quad (2.36)$$

Para o neurônio r o nível de ativação interna é

$$v_r(k) = \sum_{j=0}^q w_{rj}(k)y_j(k) \quad (2.37)$$

sendo que q denota o número total de entradas (excluindo o *threshold*) aplicadas ao neurônio r . Aqui, novamente, o peso sináptico w_{r0} é igual ao threshold θ_r aplicado ao neurônio r e a entrada y_0 correspondente é fixada no valor -1. Portanto, derivando a Equação (2.37) com relação a $y_j(k)$, encontra-se

$$\frac{\partial v_r(k)}{\partial y_j(k)} = w_{rj}(k) \quad (2.38)$$

e utilizando as equações 2.36 e 2.38 em 2.34, obtém-se a derivada parcial desejada:

$$\frac{\partial E(k)}{\partial y_j(k)} = - \sum_r e_r(k) g'_r(v_r(k)) w_{rj}(k) = - \sum_r \delta_r(k) w_{rj}(k) \quad (2.39)$$

sendo que na segunda igualdade, utiliza-se as definições do gradiente local $\delta_r(k)$ dadas na Equação (2.30) com o índice r substituindo o j .

Finalmente, usando a Equação (2.39) em (2.31), obtém-se o gradiente local $\delta_j(k)$ para o neurônio escondido j (Haykin (1994))

$$\delta_j(k) = g'_j(v_j(k)) \sum_r \delta_r(k) w_{rj}(k), \text{ neurônio } j \text{ é escondido.} \quad (2.40)$$

2.2 Planejamento de Experimentos

Pode-se definir o planejamento de experimentos (do inglês *Design of Experiments*, DOE) ou projeto experimental como sendo o projeto de qualquer experimento para coleta de informações, onde a variação está presente, quer sob o total controle do experimentador ou não. No entanto, na estatística, estes termos são geralmente utilizados para experimentos controlados. Geralmente se está interessado em avaliar o efeito de um determinado fator, ou grupo de fatores em um objeto experimental, que pode ser, uma pessoa, uma planta, um animal ou uma máquina e,

portanto, pode ser utilizada por todas as áreas de conhecimento.

Um experimento planejado é um teste, ou série de testes, no qual são feitas mudanças propositalmente nas variáveis de entrada (x_i) de um processo, de modo a se conseguir observar e identificar mudanças correspondentes na resposta (y_i).

O processo pode ser visualizado como uma combinação de máquinas, métodos e pessoas, que transforma um material de entrada em um produto, Figura 2.9. Este produto pode ter uma ou mais características de qualidade observáveis. Algumas das variáveis do processo são controláveis, enquanto outras são não controláveis. Algumas vezes, esses fatores não controláveis são chamados de fatores ruídos. Todas essas características devem ser analisadas para o planejamento do experimento, incluindo ainda a especificação do objetivo do estudo.

Figura 2.9: Processo (DOE).

Para se obter resultados confiáveis fazendo a coleta de dados no mínimo tempo e custo é de fundamental importância a utilização de técnicas estatísticas. Os experimentos planejados devem ter um objetivo claramente definido, contendo todas as variáveis importantes ao experimento e utilizando-as na escala adequada, além de terem um número razoável de experimentos, ou seja, serem suficientes para detectar até os efeitos menos significativos, e não muito numerosos a ponto de inviabilizar o estudo. Para isso é essencial o conhecimento de um especialista no assunto.

Existem dois aspectos ligados a qualquer estudo experimental: o planejamento do experimento e a análise estatística dos resultados. Eles estão intimamente relacionados, já que a técnica de análise depende diretamente do planejamento utilizado.

A utilização de um modelo estatístico para planejar e avaliar os resultados de uma investigação é uma importante ferramenta, pois proporciona ao pesquisador uma precisa interpretação do fenômeno investigado. Um experimento planejado seguindo este critério, permite que se obtenham resultados não só sobre a influência individual de cada variável estudada, mas também,

toda a gama de interações entre o total de variáveis consideradas.

Os resultados obtidos com estas experiências possuem duas aplicações imediatas: em investigações preliminares sobre certos fenômenos, onde tanto as influências individuais devidas às variáveis envolvidas, como também aquelas influências originadas da combinação de diversas variáveis precisam ser rapidamente detectadas nos programas experimentais, procurando obter modelos empíricos e semi-empíricos, em função de um conjunto de condições de operação das variáveis do processo (Costa (1997)). Sendo este último o ponto de maior interesse para este trabalho, pois pode-se com esse método, obter um modelo do sistema de estimativa de carga, fornecendo fatores e obtendo a quantidade de carga na máquina.

Foi proposto por Coleman e Montgomery (1993) as seguintes etapas para o desenvolvimento de um Planejamento de Experimentos na Indústria:

- Caracterização do problema: é absolutamente essencial desenvolver completamente todas as idéias sobre o problema e sobre os objetivos específicos do experimento. Um relato claro do problema e dos objetivos do experimento costuma contribuir substancialmente para uma melhor compreensão do processo e para uma eventual solução do problema.
- Escolha dos fatores de influência e níveis: deve-se escolher os fatores que devem variar, os intervalos sobre os quais esses fatores variarão e os níveis específicos nos quais cada rodada será feita. Exige-se conhecimento do processo para fazer isso. Esse conhecimento é em geral, uma combinação de experiência prática e conhecimento teórico. É importante investigar todos os fatores que possam ser importantes e evitar ser excessivamente influenciado pela experiência passada, particularmente nos estágios iniciais do experimento ou quando o processo não está ainda muito amadurecido.
- Seleção das variáveis de resposta: o experimentador deve ter certeza de que aquela variável realmente fornece informação útil sobre o processo em estudo. Muitas vezes, a média ou o desvio padrão (ou ambos) da característica medida será a variável-resposta. Respostas múltiplas não são raras. A capacidade do medidor é, também, um fator importante. Se a capacidade do medidor é baixa, então apenas efeitos grandes de fatores serão detectados pelo experimento, ou será necessária replicação adicional.
- Determinação de um modelo de planejamento de experimento: a escolha do planejamento envolve consideração sobre o tamanho da amostra (número de replicações), seleção de uma ordem adequada de rodadas para as tentativas experimentais, ou se a formação de blocos

ou outras restrições de aleatorização estão envolvidas. Se os primeiros passos forem feitos corretamente, este passo será relativamente fácil.

- **Condução do experimento:** quando da realização do experimento, é de vital importância monitorar o processo, para garantir que tudo esteja sendo feito de acordo com o planejamento. Erros no procedimento experimental nesse estágio, em geral, destruirão a validade do experimento. O planejamento geral, do início até o fim, é crucial para o sucesso. É fácil subestimar os aspectos logísticos e de planejamento em um ambiente industrial complexo.
- **Análise dos dados:** métodos estatísticos devem ser usados para analisar os dados, de modo que os resultados e conclusões sejam objetivos. Se o experimento foi planejado corretamente e se foi realizado de acordo com o planejamento, então o tipo de métodos estatísticos exigidos não será complicado.
- **Conclusões e recomendações:** uma vez analisados os dados, o experimento deve acarretar conclusões práticas sobre os resultados e recomendar um curso de ação. Métodos gráficos são em geral, usados nesse estágio. Sequências de acompanhamento e testes de confirmação devem ser também realizados para validar as conclusões do experimento.

Existem quatro ferramentas fundamentais no planejamento de experimento para garantir um bom resultado.

1. **Plano de teste:** a preparação dos níveis dos fatores e das unidades experimentais, objeto a ser testado, no projeto.
2. **Agrupamento planejado:** O agrupamento é uma técnica extremamente importante, utilizada com o objetivo de aumentar a precisão de um experimento. Em certos processos, pode-se controlar e avaliar, sistematicamente, a variabilidade resultante da presença de fatores conhecidos que perturbam o sistema, mas que não se tem interesse em estudá-los. O agrupamento é usado, por exemplo, quando uma determinada medida experimental é feita por duas diferentes pessoas, levando a uma possível não homogeneidade nos dados. Outro exemplo seria quando um determinado produto é produzido sob as mesmas condições operacionais, mas em diferentes bateladas. De modo a evitar a não homogeneidade, é melhor tratar cada pessoa e batelada como um bloco.
3. **Aleatorização:** Os métodos estatísticos requerem que as observações, ou os erros, sejam variáveis aleatórias distribuídas independentemente. Os experimentos, com suas réplicas,

devem ser realizados de forma aleatória, de modo a garantir a distribuição equânime de todos os fatores não considerados. Por exemplo, ao realizar-se um experimento para determinar as variáveis determinantes do acabamento da peça em uma retífica, deve-se cuidar da aleatoriedade na execução do experimento, pois fatores críticos que não estão no estudo como temperatura ambiente e lote de matéria prima, podem influenciar as variáveis de interesse de forma diferenciada, o que compromete a independência e a variabilidade entre os erros experimentais.

4. Replicação: Fazer um experimento com réplicas é muito importante por dois motivos. O primeiro é que isto permite a obtenção do erro experimental. A estimativa desse erro é básica para verificar se as diferenças observadas nos dados são estatisticamente diferentes. O segundo motivo se refere ao fato de que, se a média de uma amostra for usada para estimar o efeito de um fator no experimento, a replicação permite a obtenção de uma estimativa mais precisa desse efeito.

2.2.1 Experimentos Com Mais de Um Fator

Por causa da complexidade da maioria dos processos, geralmente são estudados vários fatores em um experimento. Uma abordagem comum, em um experimento com vários fatores, é mudar um fator de cada vez. Os experimentadores citam como razões para suportar essa abordagem, o fato de que se mais de um fator for mudado ele não conseguirá determinar qual fator foi responsável pela mudança na resposta.

Existem duas grandes deficiências no estudo de um fator de cada vez. O primeiro é que frequentemente existem interações entre fatores que estão em estudo. Uma interação significa que o efeito de um fator no resultado pode depender no nível de algum outro fator. Figure 2.10 contém gráficos respostas ilustrando vários graus de interação. Na figura (a) nota-se que não há interação entre os fatores, pois a inclinação da reta entre os valores mínimo e máximo de cada fator é a mesma, já nos outros gráficos nota-se inclinações diferentes das retas, principalmente no gráfico (d) onde a interação entre os fatores é maior, será mostrado mais adiante como elaborar esse tipo de gráfico Moen et al. (1999).

A segunda deficiência no estudo de um fator de cada vez é a ineficiência. A cada mudança de fator os dados coletados previamente são colocados de lado e novos dados são coletados. Cada conjunto de dados fornece informação de apenas um fator.

Figura 2.10: Exemplo de interações entre fatores Moen et al. (1999).

O projeto ou experimento fatorial prevê uma eficiente alternativa ao estudo de um fator de cada vez. O projeto fatorial permite o estudo de interações entre fatores e permite que toda a informação do experimento seja usada para estudar cada fator (Moen et al. (1999)).

Para se testar um experimento fatorial completo tem que se executar todas as combinações dos níveis de cada fator, por exemplo se existem 3 fatores, o primeiro com dois níveis, o segundo com 3 níveis e o terceiro com 5, então será necessário executar 30 ($2 \times 3 \times 5$) rodadas de teste, onde uma rodada é definida como o conjunto de testes necessários para se obter o resultado do teste, para uma réplica de teste. No caso de estudo com fatores de 2 níveis (mínimo e máximo) usa-se a notação 2^k , onde k é o número de fatores a serem testados, no caso de 3 fatores são necessários $2 \times 2 \times 2 = 8$ testes. Os experimentos 2^k constituem uma importante classe de planejamento, pois:

- O número de ensaios envolvidos na realização desses experimentos é relativamente pequeno.
- A região experimental nestes experimentos pode ser restrita, porém pode indicar tendências e determinar uma melhor (ou mais adequada) direção para novos experimentos.

- Estes experimentos podem ser aumentados com a inclusão de novos níveis e/ou novos fatores.
- Os experimentos 2^k formam a base dos Experimentos Fatoriais Fracionados, onde somente alguns dos possíveis tratamentos são observados.

Para se mostrar a combinação de fatores e níveis de um experimento fatorial pode-se montar uma matriz de planejamento, que pode ser montada de diversas maneiras. Em um experimento 2^k geralmente é usada a notação - (menos), para fatores no nível mínimo e + (mais) para fatores no nível máximo. Para fatores qualitativos os sinais + e - podem ser escolhidos arbitrariamente. Ver exemplo na Tabela 2.1, onde os valores da tabela não foram tirados de um exemplo real, foram escolhidos apenas para fins ilustrativos.

Tabela 2.1: Exemplo de matriz experimental.

	Fatores		Saída
Teste	A	B	Y
1	-	-	6
2	+	-	11
3	-	+	8
4	+	+	15

Pode-se calcular o efeito (EF) de cada fator da seguinte forma

$$\begin{aligned}
 EF_A &= \frac{15 + 11}{2} - \frac{6 + 8}{2} = 6 \\
 EF_B &= \frac{15 + 8}{2} - \frac{11 + 6}{2} = 3
 \end{aligned} \tag{2.41}$$

ou seja, para o fator A, a média das respostas referentes ao valor negativo de A (neste caso 6 e 8) são subtraídas da média das respostas referentes ao valor positivo de A (neste caso 15 e 11), o mesmo é feito para o fator B.

Pode-se mostrar o efeito de cada fator de forma gráfica como na Figura 2.11

Figura 2.11: Exemplo de gráfico de efeito.

Para ver uma forma gráfica da tabela 2.1 ver Figura 2.12 que é chamada de árvore amostral, sendo que para as duas formas as letras A, B, C e D representam os fatores testados.

Figura 2.12: Exemplo de árvore experimental.

Para a execução dos testes começa-se pela última linha da árvore e sobe-se até a primeira linha observando quais os níveis para cada fator. Na árvore acima começa-se pelo primeiro nível de D, que é negativo, subindo uma linha encontra-se C também negativo, na próxima linha encontra-se B negativo e na primeira linha A também é negativo, portanto para o primeiro teste deve-se ajustar todos os fatores A, B, C e D no seu nível mínimo. Deve-se repetir esse procedimento para todos os níveis da última linha.

Em um experimento fatorial completo é necessário considerar a interação de todos os fatores, no caso de um experimento com 3 fatores, por exemplo A, B e C, é necessário analisar A, B, C, AB, AC, BC e ABC. No exemplo anterior, pode-se acrescentar na tabela a interação AB multiplicando A por B.

Tabela 2.2: Exemplo de matriz experimental completa.

	Fatores			Saída
Teste	A	B	AB	Y
1	-	-	+	6
2	+	-	-	11
3	-	+	-	8
4	+	+	+	15

O desenvolvimento do modelo matemático baseado nos dados é outro método quantitativo para a análise. $Y=f(x)$ pode ser escrito como uma expressão matemática.

$$Y = \beta_0 + \beta_A A + \beta_B B + \beta_{AB} AB. \quad (2.42)$$

Os efeitos são usados para estimar os coeficientes (1/2 dos efeitos) e β_0 é a média de todas as saídas. Este modelo pode auxiliar na previsão, investigações futuras, análise de resíduos e testes confirmatórios.

Continuando o exemplo anterior:

$$\begin{aligned}
 \beta_A &= 0,5 \left(\frac{15 + 11}{2} - \frac{6 + 8}{2} \right) = 3 \\
 \beta_B &= 0,5 \left(\frac{15 + 8}{2} - \frac{11 + 6}{2} \right) = 1,5 \\
 \beta_{AB} &= 0,5 \left(\frac{15 + 6}{2} - \frac{11 + 8}{2} \right) = 0,5 \\
 \beta_0 &= \frac{15 + 11 + 8 + 6}{4} = 10 \\
 Y &= 10 + 3A + 1,5B + 0,5AB.
 \end{aligned} \quad (2.43)$$

2.2.2 Fatorial Fracionário

O número de corridas exigido por um experimento fatorial completo aumenta geometricamente com o aumento do número de fatores. Experimentos fatoriais com dois ou três fatores fazem uso eficiente dos recursos utilizando toda a informação para estimar o efeito médio e inte-

rações. Porém, quando o número de fatores aumenta, aumenta também a parte dos dados que são utilizados para estimar interações de maior ordem, que geralmente são insignificantes.

Os experimentos fatoriais fracionários são uma importante classe de planejamento de experimentos que permitem que o tamanho dos experimentos fatoriais sejam mantidos viáveis, mas ainda permitindo a estimativa de efeitos importantes (Moen et al. (1999)).

Estes experimentos fatoriais estão fundamentalmente baseados em três idéias:

1. Efeitos principais: Quando existem muitas variáveis, o sistema ou processo será dirigido por alguns efeitos principais e interações de baixa ordem.
2. Propriedades projetivas: Fatoriais fracionados podem ser projetados para que os fatores mais significantes tenham os maiores efeitos.
3. Experimentos seqüenciais: É possível combinar as corridas de dois (ou mais) fatoriais fracionados para montar sequencialmente um planejamento maior para estimar os fatores e interações de interesse (Montgomery (2001)).

A notação utilizada para um experimento fatorial fracionário leva em consideração a diferença de um fatorial completo para o fracionário, enquanto 2^7 indique um fatorial completo com 7 fatores e 128 rodadas, 2^{7-4} indica uma fração de 1/16 do fatorial completo 2^7 e resulta em 8 rodadas de testes.

2.3 Motor Síncrono Trapezoidal

O motor trifásico utilizado é ligado ao inversor na forma estrela, como mostra a (Figura 3.5), e tem o arco magnético de 180° e a corrente de fase de onda quadrada de 120° . Ligando sempre duas fases ao mesmo tempo obtém-se uma força magnetomotriz distribuída de tal forma que a qualquer momento existem dois setores, de polaridades opostas, cada um com 120° elétricos de largura, separados por setores de 60° elétricos com força magnetomotriz zero.

A (Figura 2.13) mostra um modelo simplificado deste tipo de motor, onde a área mais escura no estator representa seu polo norte com arco de 120° e a parte mais escura do rotor representa seu polo norte com arco magnético de 180° . Chaveando-se as três fases adequadamente a cada 60° obtém-se torque constante, pois, sempre $\frac{2}{3}$ do rotor vai estar sobre influência do campo magnético do estator.

Figura 2.13: Motor BPM com arco magnético de 180° e corrente de fase de onda quadrada de 120° .

As formas de onda das correntes de estator podem ser vistas na (Figura 2.14)

Figura 2.14: Forma de onda das correntes de fase.

A potência instantânea sendo convertida de elétrica para mecânica é

$$P = \omega T_e = 2eI \quad (2.44)$$

sendo I a amplitude da corrente elétrica e o número 2 vem do fato que duas chaves do inversor estão conduzindo ao mesmo tempo. Portanto o torque eletromagnético é dado por

$$T_e = 4N_{ph}B_g l r_1 I(Nm). \quad (2.45)$$

Escrevendo $E = 2e$, para representar a força eletromotriz combinada para duas fases em série, as equações de torque e força eletromotriz podem ser escritas da seguinte forma

$$E = k\phi\omega \quad (2.46)$$

$$T = k\phi I \quad (2.47)$$

sendo

$$k = \frac{4}{\pi}N_{ph} \quad (2.48)$$

$$\phi = B_g r_1 \pi l. \quad (2.49)$$

Fica claro que com formas de ondas adequadas como as da (Figura 2.14) e com comutações das chaves do inversor corretas é possível controlar o motor BPM (Miller (1993)).

As características de torque T e velocidade ω do motor BPM podem ser descritas através das seguintes equações

$$\omega = \omega_0 \left[1 - \frac{T}{T_0} \right] \quad (2.50)$$

com a velocidade em vazio sendo dada por

$$\omega_0 = \frac{V}{k\phi} \text{rad/sec} \quad (2.51)$$

e o torque do rotor travado como

$$T_0 = k\phi I_0 \quad (2.52)$$

sendo I_0 a corrente de rotor travado e é igual a V/R , sendo que R é a soma das resistências de duas fases em série e V é a tensão de alimentação.

Controlando a tensão de alimentação com um PWM pode-se controlar a corrente das fases e assim controlar o motor.

Essas características do motor serão utilizadas para calcular a quantidade de carga na lavadora de roupas, o tipo de controle implementado no microcontrolador da máquina gera um sinal PWM para controlar a velocidade do cesto, disponibilizando mais ou menos corrente ao motor para que a velocidade permaneça constante independente da quantidade de carga, com isso pode-se deduzir que a corrente é proporcional à quantidade de carga na máquina, pois é necessário mais torque para manter a máquina na velocidade determinada se a carga for maior.

2.4 Algoritmos de Estimativa de Carga

Não é comum no meio acadêmico o estudo de algoritmos de estimativa de quantidade de roupas aplicado em máquinas de lavar roupa, porém na indústria de eletrodomésticos a pesquisa deste tipo de algoritmo é bastante frequente.

Existem várias patentes sobre o assunto para máquinas de lavar. Na patente KR20050011569 (Cho Kwan et al. (2005)), uma máquina acionada por um motor DC sem escovas, tem a velocidade do cesto definida como uma primeira velocidade mais baixa onde a corrente do motor é medida, então a velocidade do motor é comandado a uma velocidade mais elevada onde a corrente do motor é medida novamente, são aplicados filtros passa-baixa com diferentes frequências de corte às correntes e com esta informação o desequilíbrio é estimado e utilizado para compensar a medição do tamanho da carga, que é feito com a amplitude da corrente.

Em JP4256788 (Nakamura e Kiuchi (1992)) uma rede neural é usada para estimar a quantidade de carga em uma máquina de lavar roupas, e o processo de aprendizagem é durante o uso da máquina, para as primeiras poucas vezes que a máquina é usada, a quantidade da carga tem que ser indicada pelo usuário, com base nestas informações a rede neural é treinada.

Baseado no conhecimento obtido desses estudos e patentes e com a experiência adquirida durante a elaboração deste trabalho, pode-se definir alguns requisitos para o algoritmo de estimativa automática de quantidade de carga em uma máquina de lavar roupas.

1. Básicos

- Durabilidade (sensor/sistema)
- Confiabilidade

- Robustez

2. Desempenho

- Precisão
- Repetibilidade

Existem alguns princípios para determinação da quantidade de roupa conhecidos, pode-se citar alguns deles:

1. Método inercial

A quantidade de energia necessária para colocar determinada massa em movimento é função do torque do motor, massa deslocada, perdas por atrito e da diferença de velocidade inicial e final. Este método descrito na patente PI0300737-5 (Martins et al. (2003)) exige alguma forma de estimar a velocidade do cesto onde a roupa é acondicionada para o processo de limpeza. Alguns fatores que influenciam neste método são o torque do motor, as perdas por atrito e o modo de determinação da velocidade do cesto, conforme descrito abaixo. O equacionamento deste método é descrito na patente citada acima.

Torque do motor - O torque do motor é função da tensão de alimentação e da velocidade do motor. Cada tipo de motor apresenta uma família de curvas características torque versus velocidade em função da tensão de alimentação.

Perda por atrito - A perda por atrito pode variar com o tempo em função do desgaste dos mecanismos (mancais, buchas, eixos) ou por mudança de propriedade física da lubrificação. O atrito pode ser estimado pelo tempo necessário para desacelerar determinada massa. Assim, quando o motor é desligado, quanto mais rápido o cesto com as roupas parar, maior é o atrito.

Velocidade do cesto - Existem inúmeras maneiras para estimar a velocidade do cesto: **Tacogerador**: Sensor elétrico de velocidade angular acoplado diretamente ao eixo de alguma parte em movimento angular sensível à velocidade do cesto móvel que comporta a roupa do consumidor. A tensão elétrica e/ou frequência do sinal gerado pelo tacogerador é função da velocidade angular. **Óptico**: Sensor óptico de velocidade angular acoplado diretamente ao eixo de alguma parte em movimento angular sensível à velocidade do cesto móvel que comporta a roupa do consumidor. A frequência de chaveamento do sensor óptico é função da velocidade angular. **Back EMF**: O motor utilizado como elemento propulsor da máquina de lavar gera sinal elétrico quando o motor é desligado. A amplitude e a frequência deste sinal elétrico é função da velocidade do cesto.

2. Método de saturação com água

Neste método, a quantidade de água necessária para molhar a roupa é função da sua quantidade e propriedades de absorção. No início do ciclo de lavagem, uma válvula permite a entrada de água até um determinado nível associado a um procedimento que molha toda a roupa no cesto, este procedimento pode consistir em rotacionar o cesto lentamente durante o enchimento de água. Desliga-se a válvula e liga-se uma bomba para recircular a água que já foi admitida ao cesto. Durante esta recirculação, mede-se a variação do nível de água e um algoritmo de decisão define o nível de líquido a ser utilizado no processo de lavagem, esse método é descrito na patente PI0601846-7 (Rocha (2008)). O nível de água é determinado por um sensor de nível, o qual converte a pressão exercida pela coluna de água no tanque em sinal elétrico. Alguns exemplos de sensores são os de princípio piezoelétrico, de extensometria (*strain gage*), semicondutor, indutivo, capacitivo ou resistivo.

3. Método de empuxo

Neste método, a quantidade de água necessária para contrabalancear o peso da roupa contra um sistema com determinada flutuação é função da quantidade de roupa sobre este sistema.

Bolsas plasticas cheias de ar são colocadas no fundo do cesto para que quando o tanque encha de água faça com que o cesto flutue, quanto maior a carga no cesto mais agua é necessária para fazer o cesto flutuar, medindo a quantidade de água necessária para a flutuação do cesto estima-se a quantidade de carga.

Capítulo 3

Sistema Utilizado

3.1 Máquina de Lavar Roupas

As máquinas de lavar podem ser divididas basicamente em dois subgrupos principais, as de eixo vertical (carregamento de carga pela parte superior da máquina) e as de eixo horizontal (carregamento de carga pela parte da frente da máquina). Dentro das máquinas de eixo vertical existem máquinas que utilizam agitadores e as que usam impulsosres, conhecidos como *impellers* na indústria, essa divisão pode ser vista na Figura 3.1, onde a linha tracejada delimita a aplicação do algoritmo neste trabalho, sendo que com algumas pequenas adaptações o algoritmo pode também ser aplicado em máquinas de eixo horizontal.

Figura 3.1: Tipos de máquinas de lavar.

A máquina onde o algoritmo foi desenvolvido é constituída por um tanque plástico suspenso por quatro varetas de suspensão, um exemplo deste tipo de suspensão pode ser visto na Figura 3.2. Dentro do tanque encontra-se um cesto de aço inoxidável com furos para possibilitar a extração de água. Quando este cesto é rotacionado em torno de seu eixo, existe também um *impeller* para agitar as roupas durante o processo de lavagem. Há também um motor DC síncrono trapezoidal (em inglês, *Brushless Permanent Magnet* ou BPM) de 42 polos, 400W e 1100 RPM que é conectado diretamente, através de um eixo ao *impeller*.

Figura 3.2: Máquina de lavar em corte.

Nas Figuras 3.3(a) e 3.3(b) pode-se ver o rotor e o estator do motor utilizado e na Figura 3.4 o estator é mostrado montado no fundo do tanque da máquina. Para centrifugar a carga, o cesto é acoplado ao eixo do motor através de um mecanismo de engate. Existem ainda uma bomba de drenagem e válvulas para o enchimento da máquina, assim como, um controle eletrônico que controla o motor, a bomba de drenagem e as válvulas de entrada de água e que contém um micro

controlador, relês e um inversor de frequência, cujo esquemático pode ser visto na Figura 3.5. Esse controle eletrônico ainda tem sensores para medir a tensão do barramento DC que alimenta o motor, o sensor de corrente do motor, os sensores de temperatura do motor e do módulo de controle do motor (ponte de IGBTs, como mostrado no esquemático do inversor).

(a) Rotor

(b) Estator

Figura 3.3: Rotor e estator do motor BPM.

Figura 3.4: Estator na máquina.

Figura 3.5: Esquemático do inversor.

3.2 Sistema de Aquisição e Tratamento de Dados

O algoritmo, que será descrito no Capítulo 4, foi implementado diretamente no *software* do controle eletrônico aproveitando o *software* existente na máquina. O *software* modificado, além de comandar as cargas da máquina para a execução do algoritmo, também lê as variáveis físicas

fornecidas pelos sensores e publica essa informação periodicamente em uma porta serial. Essa informação é lida por um computador e é salva em um arquivo do tipo *csv* (*Comma-separated values*) que pode ser visualizado em um editor de planilhas como o excel.

Diante da grande quantidade de testes, principalmente para gerar os dados de treinamento da rede neural, foi necessário criar um programa em Matlab para a leitura automática de todos os arquivos gerados.

Neste programa é informado apenas o local onde os arquivos se encontram e o nome do arquivo de saída que ele busca as informações relevantes em cada arquivo e as salva em um outro arquivo do excel (*xls*). Após a leitura de todos os arquivos e da criação do arquivo de saída algumas das variáveis são mostradas em um gráfico na interface do programa. A Figura 3.6 mostra a interface do programa.

Figura 3.6: Programa leitor de arquivos.

Para implementar e testar a rede neural foi utilizado o Toolbox Neural Network do Matlab, que pode ser acessado da janela principal com o comando “*nntool*” e que na versão 2009a do Matlab tem a seguinte interface gráfica, Figura 3.7.

Figura 3.7: Neural network toolbox.

Capítulo 4

Resultados

4.1 Algoritmo Desenvolvido

Baseando-se nas equações do motor, mostradas na seção 2.3, fica claro que o torque gerado pelo motor é proporcional à corrente aplicada nele, a Equação (2.52) mostra essa relação. Portanto a corrente do motor será utilizada como variável resposta para estimar a quantidade de carga dentro do cesto e o tempo de desaceleração do cesto também será utilizado para determinar a influência do atrito nesta variável.

A primeira versão do algoritmo era baseada na rotação do cesto com uma velocidade angular constante, uma velocidade alvo era estipulada ao motor e para manter essa velocidade, o controle aplicava uma corrente adequada baseada na resposta do sensor de velocidade, lembrando que o cesto está acoplado ao eixo do motor durante esta etapa, como descrito na seção 3.1. A corrente necessária para girar cargas maiores é também maior.

O problema desta abordagem é que, em regime permanente, a corrente necessária para girar o cesto é bastante pequena, e a diferença de corrente entre uma carga alta e uma carga baixa é menor ainda o que tornou essa abordagem inviável, pois a resolução do sensor de corrente presente no módulo de controle do motor não era capaz de diferenciar as cargas.

A segunda versão do algoritmo utiliza rampas de aceleração do cesto. Essa abordagem gera níveis de corrente muito maiores do que a abordagem anterior, pois a corrente do motor na partida é muito maior do que em regime permanente. A idéia é imprimir ao motor uma taxa de aceleração constante, com a intenção de diferenciar a quantidade de carga, pois cargas maiores exigem do motor um torque gerado maior, para uma mesma taxa de aceleração quando

comparadas com cargas menores, ou mesmo com o cesto vazio, portanto o algoritmo de controle do motor aplica diferentes níveis de corrente ao motor baseando-se na rampa de aceleração desejada e na quantidade de carga dentro da máquina de lavar.

O cesto de uma máquina pode ser considerado um cilindro, se algumas curvas em sua base são desconsideradas, portanto o momento de inércia pode ser calculado pelas equações 4.1 e 4.2 para um cilindro oco e um cilindro maciço (cesto cheio de carga) respectivamente.

$$J_c = \frac{1}{2}m(R_1^2 + R_2^2) \quad (4.1)$$

$$J_c = \frac{1}{2}mR_1^2 \quad (4.2)$$

sendo m a massa do cilindro e R_1 e R_2 os raios externo e interno, ver Figura (4.1).

O torque mecânico de aceleração do motor é proporcional ao momento de inércia do cesto como mostra a Equação (4.3)

$$T = J_c\alpha = J_c\dot{\omega} \quad (4.3)$$

com α sendo a aceleração angular.

Pode-se concluir que a distribuição da carga dentro do cesto influencia bastante no resultado. Para minimizar esta influência foi implementado no algoritmo uma centrifugação pequena antes do início das medidas para distribuir melhor as cargas, mesmo assim com o aumento da quantidade de carga, R_2 diminui e quando a máquina está com sua carga máxima o cilindro se torna maciço, o que indica que a relação carga e inércia não é linear.

Figura 4.1: Momento de inércia de um cilindro.

Outro fator que influencia a medida da corrente é o atrito, para minimizar a influência do atrito estático, que pode variar de máquina para máquina e com o tempo. O cesto é comandado para uma velocidade pequena e constante antes da rampa de aceleração constante ser iniciada.

E para minimizar a influência do atrito dinâmico é medido o tempo que o cesto demora para parar depois da rampa de aceleração, essa medida também é levada em consideração no cálculo da massa dentro do cesto.

Com as considerações feitas acima o seguinte algoritmo foi desenvolvido:

1. Acelerar o cesto até 165 RPM.
2. Desligar o motor e esperar o cesto parar de girar. (distribuir a carga)
3. Acelerar o cesto até 35 RPM.
4. Medir a tensão de linha, a temperatura do motor e a temperatura do módulo de controle do motor.
5. Acelerar o cesto até 165 RPM com uma taxa de aceleração constante.
6. Medir a corrente do motor entre as velocidades de 55 e 155 RPM com uma período de aquisição constante ($200 \mu s$) e somar toda a corrente medida.
7. Desligar o motor e medir o tempo de desaceleração.
8. Repetir os passos 3 a 7 seis vezes.
9. Calcular a média das seis medidas (para melhorar repetibilidade).

Como mencionado acima, o primeiro e segundo passos tem o objetivo de distribuir a carga uniformemente dentro do cesto. O terceiro passo serve para minimizar o efeito do atrito estático na medida, como o algoritmo é executado antes de se molhar a carga existe a possibilidade de alguma peça de roupa ser jogada para fora do cesto caso a velocidade angular seja muito elevada, por isso a escolha de uma velocidade pequena, os 165 RPM, para prevenir esse problema.

No passo 4 são feitas as medidas da tensão de barramento do inversor, da temperatura do motor e do módulo de controle do motor, isso é feito porque todos esses fatores afetam a medida de corrente e serão levadas em consideração na hora do cálculo da massa no cesto.

A resistência do enrolamento do estator varia com a temperatura, portanto a corrente também varia com a temperatura do motor, a temperatura do módulo de controle do motor faz a resistência *shunt*, o sensor de corrente, variar seu valor, também alterando a medida da corrente, e por fim a tensão do barramento também tem relação direta com a corrente do motor pela lei de Ohm.

Resumindo todos os fatores mais relevantes na leitura da corrente:

1. Distribuição da carga.
2. Atrito do sistema mecânico da máquina.
3. Tensão de barramento do inversor.
4. Temperatura do motor.
5. Temperatura do módulo de controle do motor (onde existe o sensor de corrente).

Todas essas influências têm base teórica e foram comprovadas experimentalmente, como será mostrado na seção 4.2.

A rampa de aceleração constante é comandada no passo 5, para que a corrente possa ser medida. Essa corrente é medida no passo seguinte com uma taxa de aquisição constante, e no final deste passo é feita a soma de toda a corrente medida.

O tempo que o cesto leva para desacelerar é medido no sétimo passo, esse tempo depende do atrito dinâmico do sistema e é utilizado no cálculo da quantidade de carga para eliminar a influência do atrito na medida da carga.

A algoritmo é repetido por 6 vezes para filtrar ruídos que possam eventualmente interferir nas medidas, a quantidade de repetições foi escolhida como sendo 6 para não aumentar demasiadamente o tempo do ciclo de lavagem, mas ainda permitir a filtragem adequada das medidas.

A Figura (4.2) ilustra o perfil de velocidade do cesto, e a corrente do motor. O fluxograma do algoritmo desenvolvido pode ser observado na Figura (4.3).

Figura 4.2: Comportamento de ω e I entre os passos 3 e 7.

Figura 4.3: Fluxograma do algoritmo.

As seções 4.2 e 4.3 mostram como os dados obtidos com o algoritmo são usados para a obtenção da quantidade de carga.

4.2 Planejamento de Experimentos

Com base no conhecimento adquirido sobre a máquina de lavar utilizada e em experimentos anteriormente realizados, os fatores que têm mais influência no algoritmo para estimar a sua carga foram escolhidos para compor uma nova experiência. Estes fatores são: máquina, motor, temperatura do motor, temperatura do módulo de controle do motor, placa de controle eletrônico, tamanho da carga e tensão.

Devido às variações normais nos processos de produção de peças, as máquinas podem ter massas e dimensões um pouco diferentes umas das outras, que podem mudar a inércia e atrito do sistema. Motores podem ter diferentes torques e resistência de enrolamento, que pode mudar a corrente do motor. A temperatura do motor e de seu módulo de controle são outros fatores que podem mudar a leitura da corrente, pequenas variações em componentes eletrônicos na placa de controle também podem afetar essa leitura e tensões diferentes das nominais também impactam a corrente do motor.

Se um experimento fatorial completo fosse realizado com esses sete fatores seriam necessárias $2^7 = 128$ (sete fatores com dois níveis cada) testes, ou corridas. Uma vez que é necessário executar pelo menos três vezes o experimento, a fim de se avaliar a variação dentro do conjunto acima, haveriam 384 medições para serem feitas. Para simplificar o experimento, um experimento fatorial fracionário foi projetado com $2^{7-2} = 32$ corridas e três medições para cada configuração, portanto, um total de 96 medidas foram realizadas. O planejamento experimental completo é mostrado na Tabela 4.1, e sua árvore experimental na Figura (4.4), tanto na tabela quanto na figura os fatores A, B, C, D, E, F e G representam máquina, motor, temperatura do motor, temperatura do módulo de controle do motor, placa de controle eletrônico, quantidade de carga e tensão do barramento, respectivamente, -1 e 1 representam os níveis alto e baixo para cada fator.

Como definido na seção 4.1, para cada configuração de teste as medidas são repetidas 6 vezes e o valor utilizado aqui é a média dessas medidas, portanto, na Figura (4.4) cada medida da última linha é a média das 6 medidas do algoritmo.

Com o *software JMP*, de análise estatística, foram avaliados os resultados dos testes realizados. Primeiro foi verificada a influência dos fatores avaliados nas variáveis resposta, corrente do motor e tempo de desaceleração. A influência na corrente e no tempo de desaceleração podem ser visto nas Figuras (4.5) e (4.6) respectivamente.

Tabela 4.1: Tabela experimental.

	A	B	C	D	E	F	G		A	B	C	D	E	F	G		A	B	C	D	E	F	G
1	-1	-1	-1	-1	-1	-1	1	33	-1	1	-1	1	-1	1	1	65	1	-1	1	-1	1	-1	1
2	-1	-1	-1	-1	-1	-1	1	34	-1	1	-1	1	1	-1	-1	66	1	-1	1	-1	1	-1	1
3	-1	-1	-1	-1	-1	-1	1	35	-1	1	-1	1	1	-1	-1	67	1	-1	1	1	-1	-1	1
4	-1	-1	-1	-1	1	1	-1	36	-1	1	-1	1	1	-1	-1	68	1	-1	1	1	-1	-1	1
5	-1	-1	-1	-1	1	1	-1	37	-1	1	1	-1	-1	1	-1	69	1	-1	1	1	-1	-1	1
6	-1	-1	-1	-1	1	1	-1	38	-1	1	1	-1	-1	1	-1	70	1	-1	1	1	1	1	-1
7	-1	-1	-1	1	-1	1	-1	39	-1	1	1	-1	-1	1	-1	71	1	-1	1	1	1	1	-1
8	-1	-1	-1	1	-1	1	-1	40	-1	1	1	-1	1	-1	1	72	1	-1	1	1	1	1	-1
9	-1	-1	-1	1	-1	1	-1	41	-1	1	1	-1	1	-1	1	73	1	1	-1	-1	-1	-1	1
10	-1	-1	-1	1	1	-1	1	42	-1	1	1	-1	1	-1	1	74	1	1	-1	-1	-1	-1	1
11	-1	-1	-1	1	1	-1	1	43	-1	1	1	1	-1	-1	1	75	1	1	-1	-1	-1	-1	1
12	-1	-1	-1	1	1	-1	1	44	-1	1	1	1	-1	-1	1	76	1	1	-1	-1	1	1	-1
13	-1	-1	1	-1	-1	1	1	45	-1	1	1	1	-1	-1	1	77	1	1	-1	-1	1	1	-1
14	-1	-1	1	-1	-1	1	1	46	-1	1	1	1	1	1	-1	78	1	1	-1	-1	1	1	-1
15	-1	-1	1	-1	-1	1	1	47	-1	1	1	1	1	1	-1	79	1	1	-1	1	-1	1	-1
16	-1	-1	1	-1	1	-1	-1	48	-1	1	1	1	1	1	-1	80	1	1	-1	1	-1	1	-1
17	-1	-1	1	-1	1	-1	-1	49	1	-1	-1	-1	-1	-1	-1	81	1	1	-1	1	-1	1	-1
18	-1	-1	1	-1	1	-1	-1	50	1	-1	-1	-1	-1	-1	-1	82	1	1	-1	1	1	-1	1
19	-1	-1	1	1	-1	-1	-1	51	1	-1	-1	-1	-1	-1	-1	83	1	1	-1	1	1	-1	1
20	-1	-1	1	1	-1	-1	-1	52	1	-1	-1	-1	1	1	1	84	1	1	-1	1	1	-1	1
21	-1	-1	1	1	-1	-1	-1	53	1	-1	-1	-1	1	1	1	85	1	1	1	-1	-1	1	1
22	-1	-1	1	1	1	1	1	54	1	-1	-1	-1	1	1	1	86	1	1	1	-1	-1	1	1
23	-1	-1	1	1	1	1	1	55	1	-1	-1	1	-1	1	1	87	1	1	1	-1	-1	1	1
24	-1	-1	1	1	1	1	1	56	1	-1	-1	1	-1	1	1	88	1	1	1	-1	1	-1	-1
25	-1	1	-1	-1	-1	-1	-1	57	1	-1	-1	1	-1	1	1	89	1	1	1	-1	1	-1	-1
26	-1	1	-1	-1	-1	-1	-1	58	1	-1	-1	1	1	-1	-1	90	1	1	1	-1	1	-1	-1
27	-1	1	-1	-1	-1	-1	-1	59	1	-1	-1	1	1	-1	-1	91	1	1	1	1	-1	-1	-1
28	-1	1	-1	-1	1	1	1	60	1	-1	-1	1	1	-1	-1	92	1	1	1	1	-1	-1	-1
29	-1	1	-1	-1	1	1	1	61	1	-1	1	-1	-1	1	-1	93	1	1	1	1	-1	-1	-1
30	-1	1	-1	-1	1	1	1	62	1	-1	1	-1	-1	1	-1	94	1	1	1	1	1	1	1
31	-1	1	-1	1	-1	1	1	63	1	-1	1	-1	-1	1	-1	95	1	1	1	1	1	1	1
32	-1	1	-1	1	-1	1	1	64	1	-1	1	-1	1	-1	1	96	1	1	1	1	1	1	1

Figura 4.4: Árvore experimental.

Figura 4.5: Influência dos fatores na corrente.

Figura 4.6: Influência dos fatores no tempo de desaceleração.

Os gráficos acima são denominados de pareto, na coluna *Term* estão os fatores, que podem ser individuais ou combinados, a coluna *Estimate* mostra o peso calculado para cada fator ou conjunto de fatores da coluna *Term*. Pode se notar que para as duas variáveis resposta o fator que é mais relevante é a quantidade de carga, portanto esse resultado confirma a teoria e pode-se partir para a determinação do modelo de estimativa de carga baseado nestas variáveis resposta.

Para calcular o modelo matemático $Y = f(x)$, com Y sendo a quantidade de carga e x um vetor dos fatores medidos, é necessário utilizar na equação final apenas fatores que podem ser medidos durante o funcionamento normal da máquina, portanto os fatores de variação incluídos no planejamento experimental, com a intenção de avaliar a robustez das variáveis resposta,

Figura 4.7: Gráfico pareto do fatores medidos mais relevantes.

não serão levados em consideração nesta fase de modelagem. Os fatores que serão incluídos na modelagem são os fatores C (temperatura do motor), D (temperatura do módulo de controle do motor) e G (tensão do barramento) além das variáveis resposta corrente do motor e tempo de desaceleração.

A Figura (4.7) mostra o gráfico de pareto com os fatores mais relevantes para o cálculo da variável resposta Y, onde *CurrentAverage* é a corrente do motor, *AvgRPSTemp* é a temperatura do motor, *CoastAverage* é o tempo de desaceleração, *HVDCAverage* é a tensão de barramento e *AvgMMTemp* é a temperatura do módulo de controle do motor.

É importante notar que existem interações entre 2 e 3 fatores que são significativas para o resultado, por exemplo a interação entre *CurrentAverage*, *AvgMMTemp* e *AvgRPSTemp* é o fator com a quinta maior influência no resultado da estimativa de carga. Esse resultado mostra a importância do experimento fatorial, pois se fosse avaliado um fator de cada vez, essa interação não seria descoberta e o resultado seria menos preciso.

O *software JMP* calcula os pesos de cada fator do modelo da forma mostrada na seção 2.2, esses fatores são mostrados na Figura (4.7) na coluna *Orthog Estimate* e o *off-set* de cada fator é mostrado junto com os fatores na coluna *Term*, com essas informações é possível obter o modelo $Y = f(x)$, como não serão utilizados todos os fatores mostrados na Figura (4.7) para o modelo final, por causa do custo computacional, apenas o modelo final será mostrado mais adiante. A variação do peso estimado pelo modelo comparado com o peso real pode ser visto na Figura (4.8)

Figura 4.8: Peso real x estimado para fatores mais relevantes.

O peso real (peso conhecido utilizado para obtenção dos dados) está no eixo Y e o peso estimado no eixo X .

Considerando-se apenas os 6 fatores mais relevantes para diminuir o esforço computacional na hora da implementação ainda encontra-se um resultado satisfatório, como pode ser visto comparando o fator de R^2 (Coeficiente de determinação, é uma medida de ajustamento do modelo econométrico em relação aos valores observados, o R^2 varia entre 0 e 1, indicando, em percentagem, o quanto o modelo consegue explicar os valores observados, quanto maior o R^2 , mais explicativo é o modelo) mostrado na Figura (4.10) com o da Figura (4.8).

Figura 4.9: Gráfico pareto dos 6 fatores medidos mais relevantes.

Figura 4.10: Peso real x estimado para os 6 fatores mais relevantes.

Figura 4.11: Resultado do DOE, peso real x estimado.

Com essa alteração a função $Y = f(x)$ foi obtida pelo *Software JMP*, a forma como este

Software calcula o modelo pode ser visto na seção 2.2.

$$LS = a + (b * Cur) + (c * CT) - (d * MT) - g * (Cur - e) * (CT - f) - \quad (4.4)$$

$$-i * (Cur - e) * (MDT - g) * (MT - h) + k * (CT - f) * (V - j) * (MT - h)$$

sendo LS a quantidade de carga, Cur é a corrente do motor, CT é o tempo de desaceleração, MT é a temperatura do motor, MDT é a temperatura do módulo de controle do motor, V é a tensão do barramento e $a, b, c, d, e, f, g, h, i, j$ e k são constantes que podem ser obtidas da Figura (4.9), os termos multiplicativos b, c, d, g, i , e k estão na coluna do centro da figura e os outros termos na coluna da esquerda, junto com os fatores.

A Figura (4.11) mostra a diferença da carga estimada para a carga real (medida com uma balança) e na Figura (4.12) pode-se visualizar um histograma do erro da carga estimada em relação a carga real.

Figura 4.12: Resultado do DOE, histograma.

Com o experimento fatorial realizado obteve-se um desvio padrão do erro igual a 0,669 Kg e nota-se, pelo histograma, que o erro (Estimado menos Real) do modelo é com mais frequência negativo e com concentração maior perto de -0.5Kg, ou seja o modelo está estimando carga menor do que a real.

4.3 Redes Neurais

Uma vez que, para resolver o problema de estimativa de quantidade de carga, é necessário encontrar uma função que relacione a quantidade de carga com alguns parâmetros da máquina, uma arquitetura de rede *feedforward* multi-camada foi escolhida devido à sua conhecida eficiência na aproximação de funções.

A rede neural utilizada foi o *Perceptron* multicamadas, e, após a realização de diversos testes, a configuração que produziu os melhores resultados foi uma rede de três camadas com 15 neurônios nas duas camadas escondidas com suas funções de ativação tangente hiperbolica (Equação (4.5)). Para a última camada uma função de ativação linear (Equação (4.6)) foi usada para o único neurônio nesta camada. O algoritmo de treinamento utilizado foi o *backpropagation* padrão. Uma representação da rede neural utilizada é mostrado na Figura. (2.7) mostrada na seção 2.1.

$$g(n) = \frac{(1 - e^{-2n})}{(1 + e^{-2n})} \quad (4.5)$$

$$g(n) = \begin{cases} 1, & \text{se } n > 1 \\ n, & \text{se } -1 \leq n \leq 1 \\ -1, & \text{se } n < -1. \end{cases} \quad (4.6)$$

Para obter os dados de treinamento e avaliação da rede neural vários testes foram executados, neles foram utilizadas duas máquinas diferentes, três controles eletrônicos diferentes e pesos de 0 a 8 Kg incrementados de um em um kilo, os três controles eletrônicos foram instalados em cada uma das máquinas um de cada vez. Para cada configuração foram feitas seis medidas, formando o vetor de entrada, dos mesmos parametros medidos no método anterior, temperatura do motor, temperatura do módulo de controle do motor, tensão do barramento, corrente do motor e tempo de desaceleração além dessas variáveis foi adicionado na medida o número de amostras da corrente, que é uma indicação do tempo de aceleração da máquina já que a taxa de amostragem é constante, esse fator também mostrou melhoras no resultado após a avaliação dos resultados. Foram realizados um total de 324 testes, para cada um desses testes as variáveis mencionadas acima, foram medidas 6 vezes como definido no algoritmo e o valor utilizado para cada variável é a média dessas medidas.

Para treinar a rede neural os dados foram divididos aleatoriamente em dois grupos, um de

treinamento contendo 292 vetores de entrada e um de validação contendo 32 vetores de entrada. Depois do treinamento da rede neural, os resultados vistos na Figura (4.13) mostram a validação da rede neural com os 32 vetores de entrada não utilizados no treinamento.

Figura 4.13: Resultado da rede neural, peso real x estimado sem dados do DOE.

Figura 4.14: Resultado da rede neural sem dados do DOE, histograma.

A estimativa de quantidade de carga utilizando redes neurais rendeu um desvio padrão do erro muito menor que no caso anterior, 0,289 Kg. E observa-se no histograma que na maioria dos casos o erro é positivo e próximo de 0,1 Kg.

4.4 Avaliação dos Métodos

Para confirmar que ambas, a equação gerada pelo método DOE e a rede neural são robustas às variações, todos os dados coletados foram agrupados formando um total de 420 testes. Esse grupo de dados foi testado pelos dois métodos e os resultados podem ser vistos nas Figuras (4.15) e (4.16).

Figura 4.15: Resultado da equação do método DOE com os dados da rede neural, peso real x estimado.

Figura 4.16: Resultado da rede neural, peso real x estimado com os dados do DOE

O desvio padrão para o DOE e para a NN foram 0,704 Kg e 2,117 Kg, respectivamente.

A partir desses resultados, fica claro que a rede neural não foi treinada com o conjunto correto de dados, quando as variações dos fatores testados e ruídos foram introduzidas a rede neural não foi capaz de generalizar essa informação e produziu alguns resultados inaceitáveis.

4.5 Métodos Combinados

Analisando os resultados da rede neural na seção 4.4, gerados utilizando também os valores medidos nos testes realizados no DOE, observa-se que o conjunto de medidas usado para o treinamento da rede neural não foi o mais adequado.

Observou-se que o método de planejamento de experimentos é muito útil para se encontrar os fatores que influenciam o resultado desejado e para definir quais os valores máximo e mínimo que cada variável pode assumir, ou seja, o domínio de definição do problema, porém para problema complexos e não lineares esse método não gera os resultados mais precisos.

O domínio de definição de um *Perceptron* multicamadas projetado para problemas de aproximação de funções ou de reconhecimento de padrões é definido pelos valores máximos e mínimos apresentados a rede durante o processo de treinamento. Esse tipo de rede neural desconhece o comportamento do processo fora de seus domínios de definição, portanto a utilização dos dois métodos em conjunto pode gerar melhores resultados do que cada método separadamente.

O planejamento de experimentos deve ser utilizado para definir quais fatores, e em quais níveis, devem ser levados em consideração na hora dos testes para obter o conjunto de treinamento da rede neural.

Após os testes serem realizados deve-se ter cuidado quanto a separação do conjunto de treinamento do conjunto de teste, para garantir que os valores máximo e mínimo de cada variável de entrada estejam presentes no conjunto de treinamento. Assim garante-se que se utilizou o domínio de definição correto.

Para verificar a teoria acima o conjunto de treinamento foi redefinido, dos 420 vetores de entrada disponíveis 336 foram escolhidos aleatoriamente para formar o novo conjunto de treinamento e a rede neural foi treinada com esse conjunto de dados. A Figura (4.17) mostra os resultados deste experimento quando testado com os 84 vetores de entrada não utilizados no treinamento.

Figura 4.17: Resultado final da rede neural.

Figura 4.18: Resultado final da rede neural, histograma.

Neste caso, o desvio-padrão do erro foi 0,362 Kg e o histograma mostra que a maioria dos testes tem erro próximo de 0 e 0,250 Kg

Comparando com o resultado anterior do método DOE, obteve-se uma melhora de 51 % no desvio padrão do erro, mesmo considerando todas as variações nas variáveis de entrada.

Considerando que a capacidade máxima de carga da máquina utilizada é de 10Kg e o máximo volume de água é de 147 litros, essa diferença de 0,34 Kg na estimativa da quantidade de carga, representa aproximadamente 5 litros de água a cada enchimento, considerando que os 10Kg de carga são distribuídos igualmente no cesto e o nível de água é diretamente proporcional à altura que a carga atingiu no cesto, ou seja uma carga de 5 Kg, por exemplo, estaria na metade do cesto e a quantidade de água seria $\frac{147}{2}$.

4.6 Implementação da Rede Neural

4.6.1 Implementação em MatLab

Como os resultados obtidos com a utilização da rede neural foram muito positivos, o algoritmo de treinamento da rede foi implementado no MatLab para possibilitar um melhor entendimento do funcionamento da rede neural. O algoritmo de treinamento escolhido foi a regra delta generalizada, detalhada na seção 2.1 e a rede treinada é uma rede similar àquela encontrada na seção 4.3. O software, em linguagem MatLab, do treinamento e da rede neural podem ser vistos no apêndice e um fluxograma da implementação do treinamento na Figura 4.20.

Figura 4.19: Erro quadrático médio durante treinamento.

A Figura (4.19) mostra o comportamento do erro quadratico médio durante o treinamento, que também obteve aleatoriamente 336 vetores de treinamento dentre os 420 disponíveis para serem usados neste treinamento.

Figura 4.20: Fluxograma do software de treinamento e implementação da rede neural.

Os 84 vetores não utilizados para o treinamento foram utilizados para validar a rede neural, também implementada no MatLab e mostrada acima, as Figuras (4.21) e (4.22) mostram os resultados desta implementação.

Figura 4.21: Resultado da rede neural, peso real x estimado implementada no MatLab.

Figura 4.22: Resultado da rede neural implementada no MatLab, histograma.

O desvio padrão obtido nesta implementação foi de 0,381 Kg e o erro gerado nos testes estão na maioria próximos entre $\pm 0,1$ Kg.

4.6.2 Implementação em C

Uma vez tendo implementado a rede neural no MatLab e validado o conceito a implementação da rede na aplicação foi testada, ou seja, a rede neural resultante foi implementada no microcontrolador da placa eletrônica da lavadora de roupas. Os vetores de entrada utilizados para validar essa implementação foram os mesmo utilizados para validar a implementação com o MatLab, isso foi feito para ser possível comparar os métodos. Portanto, fora erros de aproximação, os resultados devem ser iguais.

O microcontrolador utilizado foi o STM8S207RB da STMicroelectronics cuja características podem ser vistas na Figura (4.23). Sendo que a frequência do *clock* foi configurada para 16MHz.

Device	Pin count	Max. number of GPIOs (I/O)	Ext. interrupt pins	Timer CAPCOM channels	Timer complementary outputs	A/D converter channels	High sink I/Os	High density Flash program memory (bytes)	Data EEPROM (bytes)	RAM (bytes)
STM8S207RB	64	52	36	9	3	16	16	128 K	2048	6 K

Figura 4.23: Características do microcontrolador utilizado.

Duas formas de implementação das funções de ativação foram avaliadas com o intuito de verificar a melhor solução, a primeira utilizando tabelas e a segunda utilizando funções da biblioteca do micro. A implementação com tabelas consome muito mais memória do microcontrolador, mas a execução é mais rápida, já, utilizando funções a execução é mais demorada porém se utiliza muito menos memória.

Como a implementação da rede neural depende de multiplicação de matrizes, uma função que realiza essa operação foi criada, na realidade, para facilitar a manipulação dos dados no microcontrolador as matrizes foram transformadas em vetores e os números de linhas e colunas foram armazenados em outras variáveis. A função implementada recebe os vetores a serem multiplicados e o número de linhas e colunas das matrizes originais, também recebe o vetor onde

o resultado será armazenado e realiza a multiplicação como se estivesse fazendo a multiplicação de matrizes, levando em consideração o número de linhas e colunas recebidas. O código C dessa função encontra-se no apêndice.

A implementação da função de ativação foi feita de tal forma que, definindo *TANH_TABLE* (utiliza-se o comando *# ifdef* para definir essa variável) a função utiliza as tabelas para a função tangente hiperbólica e sigmoidal e se *TANH_TABLE* não é definido a função utiliza as funções da biblioteca do microcontrolador para obter a resposta.

A função com as equações da biblioteca pode ser vista no apêndice, mas a tabela não foi adicionada aqui pois consiste de 2000 elementos e seria inviável mostra-la aqui (esse valor elevado de elementos é para que a resposta se assemelhe ao máximo da função original), a diminuição do número de elementos implica diretamente na precisão da resposta. Os elementos da tabela estão entre -2 e 2 e em passos de 0.002.

Para comparar as duas formas de codificação, algumas métricas foram adotadas, essas métricas são, a utilização de memória ROM do micro (código e constantes), a utilização de memória RAM do micro, a velocidade de execução da tarefa e a precisão da resposta. As duas primeiras métricas são obtidas após a compilação do código, o compilador gera um arquivo com essa informação, a velocidade de execução foi obtida monitorando com um osciloscópio um pino específico do micro, que é ligado quando a função da rede neural é chamada e desligando esse pino quando o resultado é obtido e a precisão obtida comparando os resultados com os obtidos na implementação da rede no MatLab.

Tabela 4.2: Tabela comparativa implementação micro.

	Tabela	Função
Codigo(bytes)	1410	1225
RAM(bytes)	572	572
Constantes(bytes)	17504	1496
Tempo(ms)	280	336
Desvio Erro (Kg)	0,382	0,381

Pelo resultado das métricas na Tabela 4.2, pode se concluir que a implementação das funções de ativação através das funções da biblioteca do microcontrolador é a mais adequada para essa aplicação. A quantidade de memória utilizada foi inúmeras vezes maior quando se utilizou tabelas e a diferença no tempo de execução foi de apenas 56 ms, o que para essa aplicação não

impacta no resultado, pois o processo seguinte à estimativa de carga é o enchimento, que leva vários minutos.

A métrica mais importante é a comparação do desvio padrão do erro, nos dois casos esse valor ficou bem próximo do obtido com a implementação em MatLab, deve-se notar que no caso da implementação por funções a diferença para o desvio padrão do MatLab foi apenas na oitava casa decimal, o que é um excelente resultado.

Capítulo 5

Conclusão

A preocupação com os impactos dos seres humanos na natureza é cada vez maior, por isso, o desenvolvimento de eletrodomésticos mais eficientes é tão importante. As máquinas de lavar roupas são responsáveis por uma grande parcela do consumo de água e energia elétrica de uma residência, são em média 138 litros e 0,41KWh por ciclo nas máquinas utilizadas no Brasil, para que esses eletrodomésticos sejam mais eficientes é essencial saber a quantidade de carga que vai ser lavada para que a quantidade de água usada seja otimizada.

Neste trabalho desenvolveu-se um algoritmo de estimativa automática da quantidade de carga em uma máquina de lavar roupas, com a intenção de adequar o nível da água e os ciclos de lavagem. Dois métodos foram utilizados para resolver este problema, um baseado no planejamento de experimentos e o outro baseado em redes neurais. O planejamento de experimentos é uma técnica que é utilizada para definir quais dados, em que quantidade e em que condições devem ser coletados durante um determinado experimento. Portanto, é uma técnica de fundamental importância para a indústria, pois seu uso permite resultados mais confiáveis economizando dinheiro e tempo. O segundo método citado, as redes neurais artificiais, é extremamente versátil, podendo ser utilizado para resolver uma grande variedade de problemas, por serem modelos adaptativos treináveis, assim como o método anterior, utiliza informações obtidas empiricamente e além disso consegue inferir soluções que não foram apresentadas no processo de treinamento, tornando esse método muito robusto, algo muito buscado na indústria, principalmente em produtos de alto volume de produção que podem apresentar variações devido ao processo de produção.

Durante o desenvolvimento da rede neural observou-se que quando os dados de treinamento

da rede neural não contemplam toda as variações possíveis dos dados de entrada ela pode gerar resultados errados. A utilização do planejamento de experimentos auxilia na escolha dos fatores relevantes e de seus níveis adequados. Portanto no caso em que ambos os métodos são considerados de uma forma unificada, encontrou-se a melhor solução para este problema. O método DOE foi utilizado para descobrir todos os fatores relevantes e seus níveis adequados, que afetam a estimativa de carga e uma rede neural para estimar fatores não-linear e distúrbios, a fim de melhorar a estimativa da quantidade de carga na máquina de lavar. O método combinado teve uma resposta 51 % melhor no desvio padrão do erro do que utilizando-se apenas o planejamento de experimentos. Essa diferença representa 15 litros de água por ciclo de lavagem ou 10,9 % da média de utilização de água por ciclo.

Para-se validar os resultados encontrados, a rede neural foi implementada em um microcontrolador essa implementação final da rede neural no microcontrolador mostrou-se muito eficaz, consumindo pouca memória, com um tempo de execução aceitável e muito precisa.

Para a melhoria dos resultados, futuros estudos podem ser feitos para melhorar a resolução dos sensores, principalmente o de corrente, a utilização de sensores novos sensores para se medir variáveis não estudadas aqui, assim como novas arquiteturas de redes neurais. Outra linha de pesquisa possível é a utilização de redes neurais que possam continuar a serem ajustadas durante o vida do produto.

Referências Bibliográficas

- Calado, V. e D. C. Montgomery (2003). *Planejamento de Experimentos Usando o Statistica*. E-paper, Rio de Janeiro.
- Cho Kwan, Y., K. Kim Tae, e B. Yang Soon (2005). Method for detecting the amount of the laundry for a washing machine by a value output by averaging an electric current value for motors and eccentricity. *KR20050011569 (A)* (Patente).
- Coleman, D. E. e D. C. Montgomery (1993). A systematic approach to planning for a designed industrial experiment. *Technometrics* 35(1).
- Costa, C. E. S. (1997). O “doe” como ferramenta da qualidade para o desenvolvimento de produtos à base de solo estabilizado com rejeitos industriais, para uso na construção civil. *ENEGEP*.
- Eletrobrás (2009). Procel programa nacional de conservação de energia. <http://www.eletrobras.gov.br/ELB/procel/>. Acessado em 11/05/2009., SELO PROCEL.
- Haykin, S. (1994). *Neural Networks A Comprehensive Foundation*. Prentice Hall, New Jersey.
- Hodgkin, A. L. e A. F. Huxley (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology* 117, 500–544.
- Kandel, E. R., J. H. Schwartz, e T. M. Jessell (2000). *Principles of neural science*. McGraw-Hill, New York.
- Martins, A. L., T. C. Gross, e J. L. B. Penteado (2003). System and process for detecting a load of clothes in an automatic lundry machine. *PI0300737-5* (Patente).
- McCulloch, W. S. e W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115–133.

- Miller, T. J. E. (1993). *Brushless Permanent-Magnet and Reluctance Motor Drives*. Oxford University Press, Oxford.
- Moen, D. R., W. T. Nolan, e P. L. Provost (1999). *Quality Improvement through Planned Experimentation*. McGraw-Hill, New York.
- Montgomery, D. C. (2001). *Design and Analysis of Experiments*. Jonh Wiley & Sons, New York.
- Nakamura, T. e M. Kiuchi (1992). Cloth quantity detecting device of washing machine. *JP4256788(A)* (Patente).
- Pakula, C. e R. Stamminger (2009). Electricity and water consumption for laundry washing by washing machine worldwide. *Energy Efficiency, Springer Science+Business Media B.V.*.
- Petronilho, A. e M. A. Dias (2009). Método de estimação de quantidade de carga em máquina de lavar roupas. *PI0905317-4* (Patente).
- Petronilho, A. e M. H. Terra (2010). Automatic control applied to home appliances. *2010 IEEE Multi-Conference on System and Control, Yokohama. 1*, 1085–1092.
- Rocha, C. A. (2008). Método dinâmico para determinar automaticamente o nível de líquido de lavagem e de enxágue em uma máquina de lavar. *BRPI0601846* (Patente).
- Silva, I. N., D. H. Spatti, e R. A. Flauzino (2010). *Redes neurais artificiais: para engenharia e ciências aplicadas*. Artiber, São Paulo.
- Widrow, B. e M. E. Hoff (1960). Adaptive switching circuits. *IRE Wescon Convention Record*, 96–104.

Capítulo 6

Apêndice

6.1 Função de Treinamento no Matlab

Código para treinamento da rede neural com algoritmo *backpropagation*.

```
%Programa de treinamento pela Regra Delta;
%Inicialização randomica do vetor de pesos W:
W1 = rand(15,Tam_Dados);
W1i = W1;
W2 = rand(15,16);
W2i = W2;
W3 = rand(1,16);
W3i = W2;
%Inicialização das variáveis:
Epoca = 0;
n = 0.01;
E = 10e - 7;
EQM_ANT = 10000;
%Carrega dados de treinamento
load Training_Data
D = Alvo;
%Extrai a quantidade de treinamento
Num = size(Treina);
NumC = Num(1,2);
Num = Num(1,1);
%Adiciona um vetor de Bias na matriz de entrada
X(1,:) = -ones(1,NumC);
X(2 : Num + 1,:) = Treina;
%Função que calcula o valor do Erro Quadratico Médio inicial
EQM_ATUAL = func_eqm_tansig(D,X,NumC,W1,W2,W3);
%Loop que ajusta os valores do vetor de pesos W:
while abs(EQM_ATUAL - EQM_ANT) > E
    Epoca = Epoca + 1;
```

```

EQM_PLOT(Epoca) = EQM_ATUAL;
EQM_ANT = EQM_ATUAL;
for k = 1 : NumC
    I1 = W1 * X(:, k); % Equação 2.1
    Y1 = tansig(I1); % Equação 2.2
    Y1 = [-1; Y1];
    I2 = W2 * Y1;
    Y2 = tansig(I2);
    Y2 = [-1; Y2];
    I3 = W3 * Y2;
    Y3 = sigmf(I3, [10]);
    Delta3 = (D(k) - Y3) * (Y3 * (1 - Y3)); % Equação 2.30
    W3 = W3 + n * (Delta3) * Y2'; % Equação 2.29
    for s = 2 : size(W3, 2)
        Delta2(s - 1) = W3(s) * Delta3 * (1 - Y2(s) * Y2(s)); % Equação 2.40
    end
    Delta2 = Delta2';
    W2 = W2 + n * (Delta2) * Y1'; % Equação 2.29
    for r = 1 : size(W2, 1)
        SomaDelta2 = 0;
        for p = 1 : size(Delta2, 1)
            SomaDelta2 = SomaDelta2 + W2(r, p) * Delta2(p);
        end
        Delta1(r) = SomaDelta2 * (1 - Y1(r) * Y1(r));
    end
    Delta1 = Delta1';
    W1 = W1 + n * (Delta1) * X(:, k)'; % Equação 2.29
    clear Delta1 Delta2 Delta3
end
%Função que calcula o valor do Erro Quadrático Médio
EQM_ATUAL = func_eqm_tansig(D, X, NumC, W1, W2, W3);
end
%Apresentação dos resultados:
W1f = W1;
W2f = W2;
W3f = W3;
%Chama a rede neural implementada PMC_tansig

```

6.2 Função do erro quadratico médio no Matlab

A função que calcula o erro quadratico médio é mostrada a seguir.

```

function[EQM] = func_eqm(De, Xs, Numer, W1u, W2u, W3u)
Soma2 = 0;
for i = 1 : Numer

```

```

    Ii1 = W1u * Xs(:, i);
    Yi1 = tansig(Ii1);
    Yi1 = [-1; Yi1];
    Ii2 = W2u * Yi1;
    Yi2 = tansig(Ii2);
    Yi2 = [-1; Yi2];
    Ii3 = W3u * Yi2;
    Yi3 = sigmf(Ii3, [10]);
    EQM = (De(i) - Yi3)^2;
    Soma2 = Soma2 + EQM;
end
EQM = Soma2;
EQM = EQM/Numer;

```

6.3 Rede neural implementada no Matlab

```

% Carregando matriz de amostras
Num = size(Treina_test);
Num = Num(1, 2);
% Extrair quantidade de amostras
X = [-1 * ones(1, Num); Treina_test];
for k = 1 : Num
    I1 = W1 * X(:, k);
    Y1 = tansig(I1);
    Y1 = [-1; Y1];
    I2 = W2 * Y1;
    Y2 = tansig(I2);
    Y2 = [-1; Y2];
    I3 = W3 * Y2;
    Y3 = sigmf(I3, [1 0]);
    Resultado(k) = Y3;
    Erro(k) = Y3 - Alvo_test(k);
end
% Calculo do desvio padrão Desv_pad = std(Erro) * 8

```

6.4 Função de multiplicação de matrizes em C

```

MULTIPLICATION_STATUS_TYPE Neural_Network_MultMatrix(float *matrix1,
unsigned char num_lin1, unsigned char num_col1, float *matrix2, unsigned char num_lin2,
unsigned char num_col2, float *matrix_res)
{
    unsigned char j;

```

```
    unsigned char k;
    unsigned char index_a;
    unsigned char index_b;
    float aux_1;
    float aux_2;
    MULTIPLICATION_STATUS_TYPE mult_status;
    if(I_Mat < num_lin1)
    {
        for(j = 0; j < (num_col2); j++)
        {
            aux_1 = 0;
            aux_2 = 0;
            for(k = 0; k < (num_col1); k++)
            {
                index_a = (I_Mat * num_col1);
                index_b = (k * num_col2);
                aux_2 = aux_1;
                aux_1 = (float)((((float)matrix1[index_a + k]) * ((float)matrix2[index_b + j])));
                aux_1 = aux_1 + aux_2;
            }
            matrix_res[I_Mat + j] = (float)aux_1;
        }
        mult_status = MULTIPLICATION_NOT_COMPLETED;
        I_Mat++;
        if(I_Mat == num_lin1)
        {
            mult_status = MULTIPLICATION_DONE;
        }
        else
        {
            mult_status = MULTIPLICATION_DONE;
        }
        if(num_col1 != num_lin2)
        {
            mult_status = MULTIPLICATION_INVALID;
        }
        return(mult_status);
    }
}
```

6.5 Função de ativação implementada em C

```
void Neural_Network__SigmoidFunction(float *value, unsigned char size, unsigned char
func_type)
```

```

{
    unsigned char i;
    # ifdef TANH_TABLE
        unsigned short temp;
        for(i = 0; i < size; i++)
        {
            if(func_type == TANH_TYPE)
            {
                if(value[i] <= -2)
                {
                    value[i] = -1;
                }
                else if(value[i] >= 2)
                {
                    value[i] = 1;
                }
            }
            else
            {
                temp = (unsigned short)(value[i] * 500 + 1000); // Add 1000 to turn value to
positive since it will be an index for a table
                value[i] = Tanh_Table[temp]; // Find Value in the Table.
            }
        }
    }
    else
    {
        if(value[i] <= -4)
        {
            value[i] = 0;
        }
        else if(value[i] >= 4)
        {
            value[i] = 1;
        }
        else
        {
            temp = (unsigned short)(value[i] * 250 + 1000); // Add 1000 to turn value to
positive since it will be an index for a table
            value[i] = Sigmf_Table[temp];
        }
    }
}
}
# else
    double temp;
    for(i = 0; i < size; i++)
    {
        temp = (double)value[i];

```

```
    if(func_type == TANH_TYPE)
    {
        value[i] = (float)(tanh(temp));
    }
    else
    {
        temp = (1/(1 + (exp(-temp))));
        value[i] = (float)(temp);
    }
}
#endif
}
```