UM SISTEMA DE RECONHECIMENTO DE FALA PARA ENSINO DE ROBÓTICA DESTINADO AO CONTROLE DE ROBÔS LEGO® NXT 2.0

A SPEECH RECOGNITION SYSTEM FOR TEACHING OF ROBOTICS FOR THE CONTROL OF LEGO® NXT 2.0 ROBOT

PAGANI, Diego Henrique [1]; **POSTAL**, Adriana [2]; **CASTRO**, Josúe Pereira de [3]

[1] Mestrando em Informática pela Universidade Federal do Paraná (UFPR)
dhpagani@gmail.com

[2] Professora Assistente da Universidade Estadual do Oeste do Paraná - UNIOESTE, Campus Cascavel
adriana.postal@unioeste.br

[3] Professor Assistente da Universidade Estadual do Oeste do Paraná - UNIOESTE, Campus Cascavel
josue.castro@unioeste.br

Resumo

Este trabalho apresenta o desenvolvimento de um sistema de comando por voz para robôs Lego® Mindstorms® NXT 2.0 utilizando reconhecimento de fala. Até o momento, o sistema de reconhecimento de fala está identificando os comandos emitidos pela voz de forma aceitável, com 96% no melhor caso, e 42% no pior. O projeto atualmente encontra-se em fase de desenvolvimento, tanto para melhorar o seu desempenho quanto para ampliar io número de comandos reconhecidos.

Palavras-chave: Robótica, reconhecimento de fala, Lego® NXT 2.0

Abstract

This paper presents the development and implementation of a voice command system for LEGO® MINDSTORMS® NXT 2.0 robots using speech recognition. At this moment the system is running with efficiency of 96% in the best case and 42% in the worst case. The project is under development, aiming to improve performance and to ampliate the number of recognized commands.

Key-words: Robotics, speech recognition, Lego® NXT 2.0.

INTRODUÇÃO

Com a tecnologia atual, a interatividade dos equipamentos eletrônicos vem crescendo bastante. Com cada vez mais poder computacional, smartphones, video-games, mais recentemente os televisores, e outros dispositivos, vem apresentando funcionalidades cada vez mais interessantes para o usuário, com o uso de filtros para fotos, realidade aumentada, reconhecimento de comandos por voz.

Três dos serviços mais conhecidos para reconhecimento de voz em celulares, Google Now (Google, 2014) para sistemas operacionais Android®, Cortana (Microsoft, 2014) para sistemas operacionais Windows Phone® e Siri (APPLE, 2014) para os sistemas operacionais iOS® têm como objetivo serem assistentes pessoais, capazes de ouvir comandos e executarem ações perante estas entradas. Eles não precisam de um treinamento de locutor para interpretar os comandos ditos e estão presentes na maioria dos smartphones modernos.

Visando o auxílio no ensino de robótica, este trabalho apresenta a implementação de um sistema de reconhecimento de comandos de voz, para controle de um robô Lego® Mindstorms® NXT 2.0 (LEGO, 2014). Por este conjunto de robô ser composto por diversas peças, motores e sensores, temos diversas possibilidades de combinação de peças, sendo que cada uma delas pode desempenhar diferentes ações. Com isso, os alunos podem aprender de forma prática a montar uma estrutura de um robô que seja minimamente estável e capaz de realizar certas ações. Por exemplo, podem juntar certas peças e motores e formar um carro, que se move para frente ou para trás e pelos lados. Ou talvez um sistema de segurança, onde o sensor de ultrassom pode detectar a presença de algum objeto nas proximidades, e realizar alguma ação contra este objeto. Isso exercita a imaginação, faz a prática da lógica de programação e melhora a iteração humano-robô.

O sistema foi desenvolvido na plataforma MATLAB®, e estabelece a conexão entre computador e robô através da biblioteca RWTH-Mindstorms NXT Toolbox (LFB LEHRSTUHL FÜR BILDVERARBEITUNG, 2014) por meio de conexão Bluetooth®. Optou-se pela implementação em MATLAB®, pois este software contém uma série de bibliotecas, como a Signal Analysis Toolbox que implementa todas as funções básicas de processamento de sinais e também por oferecer uma linguagem interpretada, que possibilita um processo de prototipação de sistemas mais eficiente. Além disso, facilita a descoberta de erros no código e também permite a realização de testes em tempo real através de seu console.

O trabalho está assim dividido: na seção 2 descrevemos o sistema desenvolvido, juntamente a interface; a seção 3 descreve o sistema de reconhecimento de voz e seus módulos; a seção 4 mostra a configuração dos testes; a seção 5 mostra os resultados encontrados; por fim, na seção 6, são descritas as considerações finais sobre o trabalho.

O SISTEMA

O sistema está sendo realizado de forma modular, o que reduz o custo de manutenção do código por manter o mínimo de replicação de código possível, utilizando nomes de funções e variáveis com fácil identificação de sua função.

Cada módulo é independente, unindo suas funções e comunicação pela interface do sistema, o que possibilita uma maior autonomia de cada módulo e a manutenção desta aplicação menos custosa. Abaixo segue a lista dos módulos e suas respectivas descrições. A Figura 1 mostra a interface gráfica do sistema.

• Interface: responsável por interagir com o usuário e realizar a ligação entre os outros módulos;
• Manipulação do Lego: módulo responsável por comunicar-se com o robô Lego® Mindstorms® NXT 2.0, utilizando a biblioteca RWTH-Mindstorms NXT Toolbox;
• Sistema de reconhecimento de fala: responsável por identificar o comando enviado pelo usuário.

O MATLAB® possui um sistema de criação de interfaces próprio, capaz de suprir as necessidades do desenvolvimento desde trabalho. Ele apresenta uma fácil manipulação, e um conhecimento básico de seus comandos. Por tratar todos os eventos dos componentes da tela como funções, determinar as ações destes componentes não apresentou problemas.

Figura 1 – Tela do sistema de controle

FUNCIONAMENTO DO SISTEMA

Podemos ver na Figura 1 que a interface do sistema é dividida em 4 partes: Conexão LEGO; Comandos Cadastrados; Controle Manual; Controle por voz. Abaixo, segue a descrição de cada um destes comandos.

1. Conexão LEGO: Mostra o estado da conexão com o robô, com botões de conectar ou desconectar com o robô, e com um campo de texto para inserir o nome dado ao robô nas configurações;
2. Controle Manual: Esta seção mostra os botões para controle manual do robô, indicado pelos botões coloridos. Os comandos pelos botões é: azul – frente, amarelo – girar à esquerda, laranja – girar à direita, branco – ir para trás, vermelho – parar). Ao clicar no botão, é mandado o comando ao robô e este só irá parar de executar o comando ao clicar no botão vermelho, ou seja, ao clicar no botão amarelo o robô iniciará a manobra de girar à esquerda, e se nada o interromper, ele girará à esquerda até a carga da sua bateria se esgotar.

3. Comandos cadastrados: Os comandos cadastrados são fixados dentro do código, pois precisam ser atrelados as ações do robô. É possível adicionar novas amostras aos comandos já criados para aumentar o tamanho da base de dados.

4. Controle por voz: composta por um único botão, é responsável por receber o comando dito, interpretar e enviar o comando ao robô, caso ele esteja conectado.

A princípio, é necessário que se clique manualmente no botão "Enviar comando" para que o sistema capture o som pelo microfone e então interprete o comando e realize-o. Nos trabalhos futuros será alterado esta parte para que ele sempre esteja capturando sons pelo microfone, tornando o uso do sistema intuitivo. Desta forma cria-se a necessidade de aprimorar, não só o reconhecimento de fala, como o tratamento de ruídos e também para a distinção entre comandos destinados ao robô e diálogos que gerem ruídos capazes de serem identificados como comando.

Sistema de Reconhecimento de Comandos de Voz

O sistema de reconhecimento de comandos de voz desenvolvido é dividido em módulos de gravação, pré-processamento, classificação e análise. Cada módulo é responsável por realizar uma parte específica do sistema.

MÓDULO DE GRAVAÇÃO

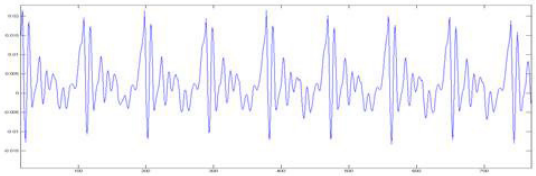
O módulo de gravação é responsável por captar amostras dos comandos, ditos pausadamente pelo locutor, e armazenar estes dados para uso futuro.

MÓDULO DE PRÉ-PROCESSAMENTO

O modulo de pré-processamento é responsável por permitir que a amostra de som seja transformada de forma que permita ao sistema reconhecer o comando emitido. Esta transformação é necessária, pois para o reconhecedor de voz, a informação mais importante é a forma em que o espectro da voz muda conforme o tempo (PLANNERER, 2005).

A Figura 2 mostra as ondulações provenientes dos ruídos ao capturarmos o espectro puro da vogal /a:/ . Como o objetivo do reconhecedor de voz é entender o que foi dito, estas ondulações ruidosas devem ser separadas.

Figura 2 – Amostra da vogal /a:/ (frequência= 11000 Hz, tempo=72ms)



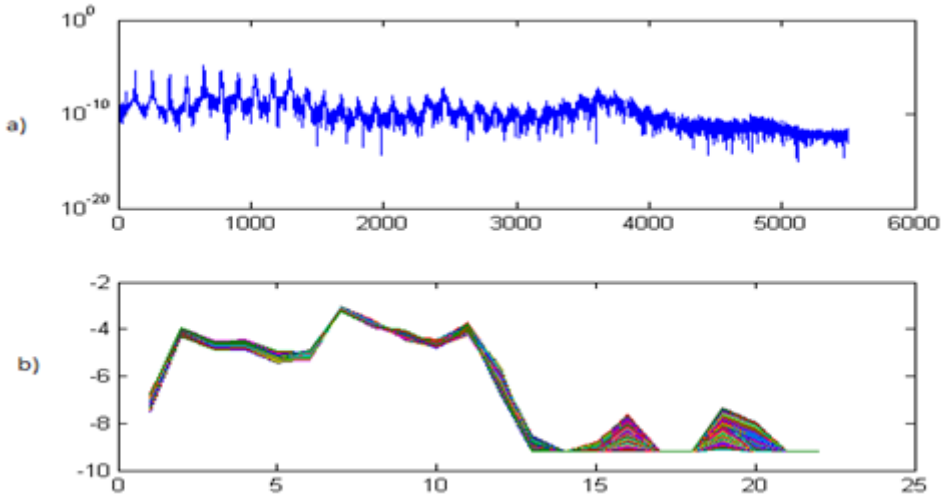
Uma das formas de contornar isto é utilizar o método Mel Frequency Cepstral Coefficients (MFCC) (SAHIDULLAH, 2012) , que filtra e normaliza o sinal, deixando as mudanças na voz mais salientes. O resultado desta operação mostra os níveis de energia contidos em cada frequência, pelo tempo igualmente espaçado. Segundo Rabiner (1993),

experimentos mostram que a percepção humana de componentes de frequência de sons não segue uma escala linear. Então para cada frequência f na escala linear, mapeia-se para a escala MEL que corresponde a real percepção do sistema auditivo humano. A Figura 3 mostra as diferenças antes e após o processamento do sinal da voz.

Figura 3 –

a) amostra da vogal /a:/ (frequência= 11000 Hz, tempo=72ms)]Log power spectrum da vogal /a:/ (frequência = 11000Hz e N = 512) e

b) amostra da vogal /a:/ após todo o processo MFCC, e aplicação dos níveis de energia (número de canais = 22).



A Figura 4 (abaixo) apresenta quatro amostras do comando “Pare” pré-processadas. Todas são semelhantes ao ouvido humano, mas os níveis de energia representados são diferentes, devido ao ruído existente ou alterações na intensidade da voz do locutor. Todas tem maior intensidade na parte esquerda do gráfico, que se assemelha ao fato da palavra “Pare” ser paroxítona.

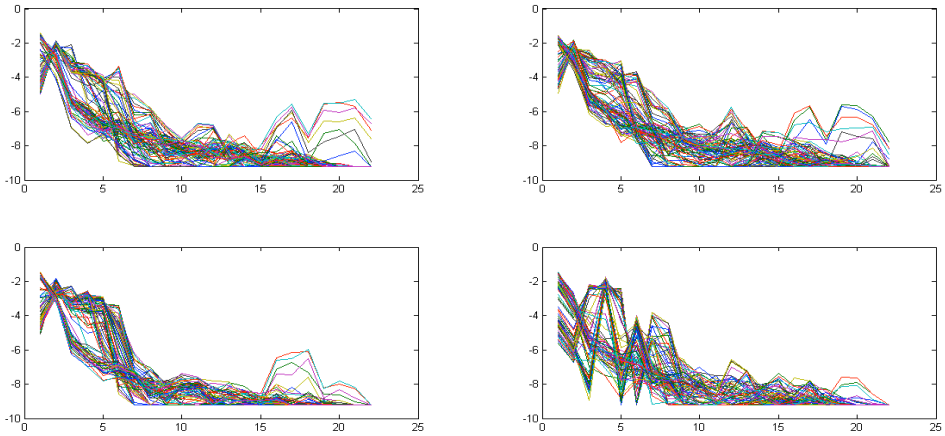


Figura 4 – 4 amostras do comando “Pare”, pós processamento

MÓDULO DE CLASSIFICAÇÃO

Este módulo é responsável por requisitar ao módulo de pré-processamento os resultados aplicados à base de dados criada pelo módulo de gravação. Então com a base de dados criada, é necessário utilizar um classificador. A princípio, testou-se com dois modelos simples de classificação: Redes Neurais Artificiais (RNA) e distância de Levenshtein.

A distância de Levenshtein (SAMWORTH, 2013) calcula, entre dois vetores (V1 e V2, o número mínimo de operações necessárias para transformar V2 em V1, adaptada para trabalhar com matrizes, o que não exige nenhum tipo de treinamento ou processamento prévio à base de dados. A Figura 5 exemplifica a ideia do algoritmo. A variável v1 apresenta uma palavra com 8 caracteres, e a v2 com 7. A diferença entre duas letras caracteriza mais duas mudanças neste vetor, tendo como resultado final 3 alterações entre estes vetores: duas de alteração e uma de eliminação.

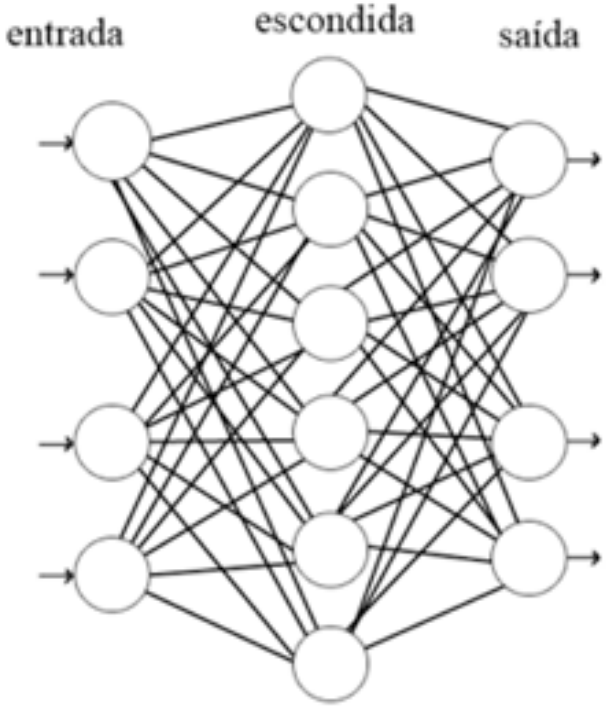
Figura 5 – Exemplo de como é calculada a distância de Levenshtein

v1 =	C	A	R	O	L	I	N	A
v2 =	K	A	R	O	L	Y	N	

Levenshtein(v1,v2) = 3

As Redes Neurais Artificiais possuem diversos tipos de organização, funcionamento e aplicações. São modelos computacionais inspirados no sistema nervoso de um animal. Este tipo de classificador, necessita de um treinamento supervisionado, para que seja possível de classificar corretamente padrões previamente apresentados a rede. A RNA implementada foi a do tipo multilayer-perceptron, construída com a toolbox do MATLAB® com os parâmetros padrão. A Figura 6, por Pagani (2012), mostra um exemplo de uma rede neural com 4 neurônios de entrada, 6 da camada intermediária e 4 de saída.

Figura 6 – Exemplo de uma Rede Neural Artificial



MÓDULO DE ANÁLISE

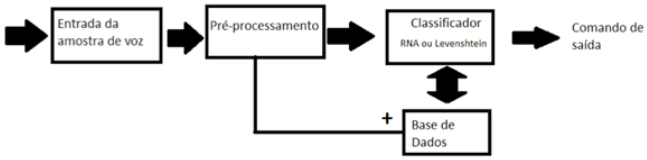
Este módulo analisa a amostra de voz do usuário, aplica o pré-processameno e identifica qual o comando a amostra de áudio representa. Isto é feito utilizando os algoritmos apresentados na sessão 3.3.

O comando identificado é então convertido em instruções de movimentação e enviado ao robô Lego® pelo computador onde o sistema

de reconhecimento de voz está sendo executado, através de uma conexão Bluetooth®.

A Figura 7 exemplifica o funcionamento geral do sistema de reconhecimento de voz. A voz é capturada, passa por um pré-processamento. Se a base de dados está sendo criada, então a voz é armazenada na base com o comando dito inserido manualmente. Se Está sendo classificada, então a matriz de saída vai para o classificador que consulta a base de dados (para a RNA, é feito o treinamento da rede, uma única vez) e a partir destes dados diz qual o comando de saída, encaminhando para o robô.

Figura 7 – Exemplo de funcionamento do sistema de reconhecimento de voz



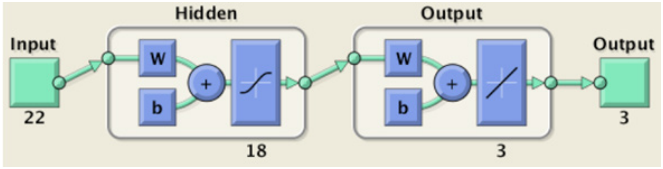
EXPERIMENTOS

Para descobrir quão preciso é o sistema de reconhecimento de comandos de voz proposto neste trabalho, conduziu-se alguns experimentos envolvendo dois locutores do sexo masculino, com três comandos: “Pare”, “Ande” e “Gire”. Para cada locutor, foi elaborada uma base de dados com 10 amostras de cada comando, totalizando duas bases (Base A do Locutor A e Base B do Locutor B). Para realizar os testes, definiu-se que cada locutor deveria fornecer 33 amostras para cada comando. Na captura de todas as amostras, definiu-se que seria realizada em apenas um canal (Mono), com 16 bits de representação e com 11000 Hz.

Devido a cada amostra gerar uma matriz de 96x22, totalizando 2112 elementos, uma rede neural com esta quantidade de neurônios de entrada necessita de um poder computacional superior ao disponível, então de maneira empírica definiu-se agrupar as matrizes por linhas, calculando a média delas, gerando um vetor resultante de 22 posições.

Com esta simplificação da dimensão, pode-se testar ambos os classificadores e avaliar seu comportamento. Com isto, a RNA possui 22 entradas, 18 neurônios na camada intermediária e 3 neurônios de saída (um neurônio é disparado para cada saída). A saída final de cada neurônio é arredondada de forma tradicional, após cada teste. A Figura 8 exemplifica a configuração da rede.

Figura 8 – Exemplo da rede neural, extraído usando a função view do MATLAB®



RESULTADOS OBTIDOS

Analisando os dados obtidos, a Tabela 1 mostra as taxas de acerto, utilizando a RNA como método de classificação. As bases de dados e as amostras foram nomeadas como A - Locutor 1 e B – Locutor 2. Foram realizados, com os 2 locutores e diferentes combinações com as bases de dados. Podemos notar que o Locutor A obteve uma taxa de acerto satisfatória utilizando sua base de dados, o que significa que a rede conseguiu aprender o padrão. O Locutor B apresentou resultados inferiores

ao do Locutor A, mesmo utilizando a sua própria base dados.

Tabela 1 – RNA com agrupamento

RNA	Agrupamento por média		
	Base LA	Base LB	Base A+B
Locutor A	78,79%	41,41%	79,79%
Locutor B	6,00%	52,53%	39,39%

A Tabela 2 mostra os resultados dos testes aplicados à base, utilizando a Distância de Levenshtein como método classificatório. Os resultados podem ser considerados melhores comparados com os resultados dos testes utilizando a RNA, pois em todos os testes a taxa de acerto foi superior.

O Locutor A se manteve com uma taxa de acerto satisfatória com qualquer base de dados, mas o Locutor B teve uma taxa de acerto inferior ao esperado utilizando a base de dados do Locutor A e a base dos dois locutores unidas.

Tabela 2 – Levenshtein com agrupamento

Distância Levenshtein	Agrupamento por média		
	Base LA	Base LB	Base A+B
Locutor A	83,84%	73,74%	83,84%
Locutor B	42,42%	73,74%	50,51%

A Tabela 3 mostra as taxas relativas aos dados sem o agrupamento por média, considerando toda a matriz 96x22.

Pode-se perceber que esses testes apresentaram resultados melhores comparado com a Tabela 2. Isso pode ter acontecido devido a perda de precisão que o agrupamento por média pode causar.

Tabela 3 – Levenshtein sem nenhum tratamento

Distância Levenshtein	Sem tratamento		
	Base LA	Base LB	Base A+B
Locutor A	95,96%	84,85%	95,96%
Locutor B	49,49%	72,73%	74,75%

CONSIDERAÇÕES FINAIS

Os testes realizados são promissores, mas o sistema ainda necessita de mais ajustes e testes. O sistema de classificação utilizando o algoritmo de Levenshtein tem bons resultados se a amostra de voz for semelhante às cadastradas no banco de dados. Já com a RNA, por ter sido realizado o agrupamento pela média, pode ter causado perda de precisão, que pode ter sido a causa do resultado inferior ao do algoritmo de Levenshtein. Existindo divergências, o sistema pode ser incapaz de reconhecer comandos incorretamente de uma voz diferente das cadastradas.

O algoritmo de Levenshtein tem uma característica que quanto maior a base, maior o tempo necessário para o reconhecimento do comando dito, pois a amostra a ser reconhecida precisa ser comparada com cada elemento presente na base de dados. Estas amostras, por serem matriciais, o custo computacional é elevado, se feito de forma tradicional. Utilizar instruções que utilizam os registradores de cálculo vetorial da CPU (Central Processing Unit) ou instruir a GPU (Graphics Processing Unit) para realizar estes cálculos pode tornar o tempo de processamento inferior ao atual.

Na RNA, por necessitar de uma etapa de treinamento em que não há garantias de

convergência, não se pode inferir nenhuma medida de tempo necessária para o treinamento, apenas limitando a quantidade de épocas. Para o reconhecimento, pós-treinamento, o tempo pode ser considerado rápido.

Outro fator que deve ser revisto, é a da captura de som para a geração da base de dados e dos testes. Temos muitos equipamentos disponíveis no mercado, em que cada um apresenta diferentes níveis de ruído e também diferentes locais que podem ser gerados as bases de dados. Estabelecer um critério de seleção de equipamentos, pesquisar formas de tratamento de ruídos externos podem ser necessários para tornar este sistema mais genérico.

REFERÊNCIAS

APPLE. Siri. Disponível em <https://www.apple.com/ios/siri/>. Acesso em 20 de agosto de 2014.

GOOGLE, Google Now. Disponível em: <http://www.google.com/landing/now/>. Acesso em 20 de agosto de 2014.

LEGO. LEGO.com Mindstorms. Disponível em: <http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com>. Acesso em: 04 abr. 2014.

LFB LEHRSTUHL FÜR BILDVERARBEITUNG. RWTH-Mindstorms NXT Toolbox. Disponível em: <http://www.mindstorms.rwth-aachen.de/trac>. Acesso em: 04 abr. 2014.

MICROSOFT. Cortana. Disponível em: <http://www.windowsphone.com/pt-br/how-to/wp8/cortana/meet-cortana>. Acesso em 20 de agosto de 2014.

PAGANI, D.H. Estudo Comparativo entre redes neurais artificiais e redes neurais pulsadas usando MATLAB. 2012. Monografia. Universidade Estadual do Oeste do Paraná.

PLANNERER, B.. An Introduction to Speech Recognition. Munich: IEEE, 2005. 69 p.

RABINER, Lawrence; JUANG, Biing-hwang. Fundamentals of speech recognition. New Jersey: Prentice Hall, 1993.

SAHIDULLAH, Md., SAHA, Goutam. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. Speech Communication, v. 54, n.4, p.543-565, 25 nov. 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167639311001622>. Acesso em 02 abr. 2014.

SAMWORTH, Richard J.. Optimal weighted nearest neighbour classifiers. The Annals Of Statistics, [S. l.], v. 40, n. 5, p.2733-2763, 18 fev. 2013. Disponível em: <http://arxiv.org/pdf/1101.5783v3.pdf>. Acesso em: 04 abr. 2014.

Artigo submetido em: 31.08.2014

Artigo aceito para publicação em: 27.12.2014