Métodos de Classificação baseado em Redes Bayesiana e Neural para Reconhecimento de Atividade Humana

Leopoldo Marchiori Rodrigues
Instituto Federal do Espírito Santo (IFES)
Vitória, Espírito Santo, Brasil
leopoldomarchiori@gmail.com

Mário Mestria
Instituto Federal do Espírito Santo (IFES)
Vitória, Espírito Santo, Brasil
mmestria@ifes.edu.br, mmestria@uol.com.br

***Resumo*— O Reconhecimento de Atividade Humana é uma área inerente à sensibilidade ao contexto, sendo aplicado, por exemplo, nas áreas de saúde, esportes e segurança. Utilizando sensores externos, como câmeras, e embarcados, como acelerômetros, existem trabalhos focados em múltiplos sensores simultaneamente auxiliando na coleta de informações em tempo real, e trabalhos dedicados a classificar a atividade eficazmente a partir de grande volume de dados. Neste trabalho foram utilizados os classificadores Rede Bayesiana e Rede Neural, construídos a partir de dados públicos, referentes à coleta de acelerômetros vestidos por voluntários. Com conhecimentos de Mineração de Dados, foi feita a seleção de atributos, construção, treinamento e testes, obtendo-se Matrizes de Confusão, taxas de acerto e tempos de processamento. O objetivo é comparar diferentes métodos de classificação com diferentes técnicas de construção e validação, visando maior adequação aos dados, avaliando-se os impactos no desempenho e nas taxas de acerto. A implementação foi feita com biblioteca pública de Mineração de Dados Weka e programação Java. Com menores tempos de processamento, a Rede Neural apresentou-se excelente. A Rede Bayesiana requereu maior investigação na configuração e técnicas de validação, atingiu os maiores tempos de processamento e obteve o melhor resultado com a técnica de validação 10-*fold cross-validation*.**

Palavras-chave—Reconhecimento de Atividade Humana; Mineração de Dados; Classificadores; Rede Bayesiana; Rede Neural

I. INTRODUÇÃO

É fato que as aplicações de tecnologia de computação vestível, com a disponibilidade de diversos tipos de serviços, têm se intensificado nos últimos anos. Por exemplo, grandes fabricantes de *gadgets*, tais como Apple e Samsung, têm lançado relógios dedicados com sensor de batimento cardíaco e outras facilidades que auxiliam o usuário nas comunicações e monitoramento da sua saúde, o que evidencia uma tendência de prover cada vez mais dispositivos vestíveis com serviços inteligentes em tempo real baseado no próprio comportamento do usuário. O termo Reconhecimento de Atividade Humana, o qual será referenciado a partir de agora como HAR (*Human Activity Recognition*), remete a uma solução, a qual ultimamente tem sido foco de intensas pesquisas, que fornece a informação em tempo real da atividade humana (classificação) baseado em dados coletados de sensores. A entrega de serviço está na linha de trabalho de Sensibilidade ao Contexto (*Context Awereness*), em que a partir da classificação, mais alguma outra informação relevante (por exemplo, a localização), fornece-se um serviço de grande utilidade [1]. A atividade é elemento chave para o contexto.

As aplicações práticas para HAR são possíveis de serem identificadas em: monitoramento de atividades físicas, saúde assistida [1], detecção de quedas [2], assistência a idosos, serviços móveis baseados em localização, segurança [3], jogos (*Nintendo Wii* ou *Kinect*) ou ainda nos esportes.

As técnicas de Mineração de Dados têm sido cada vez mais aplicadas em diversas áreas de conhecimento e tecnologias para descoberta de conhecimento a partir de grande massa de dados [4]. É um processo computacional de descoberta de padrões em grande massa de dados, transformando isso em uma estrutura inteligente e compreensível. Em processos de Classificação e de Clusterização analisa-se um conjunto de dados e gera-se um conjunto de regras, utilizando uma base de dados para treinamento, que podem ser utilizadas para classificar automaticamente novos dados de entrada. O objetivo de um algoritmo de Classificação é gerar os resultados mais corretos e precisos quanto for possível.

A Mineração de Dados e suas técnicas de classificação têm sido extensivamente utilizadas para processar os dados coletados de sensores, como, por exemplo, acelerômetros, e em seguida obter resultados úteis de classificação nas aplicações de HAR. Na Fig. 1 é mostrado um diagrama esclarecedor da contextualização do assunto a ser tratado neste trabalho.

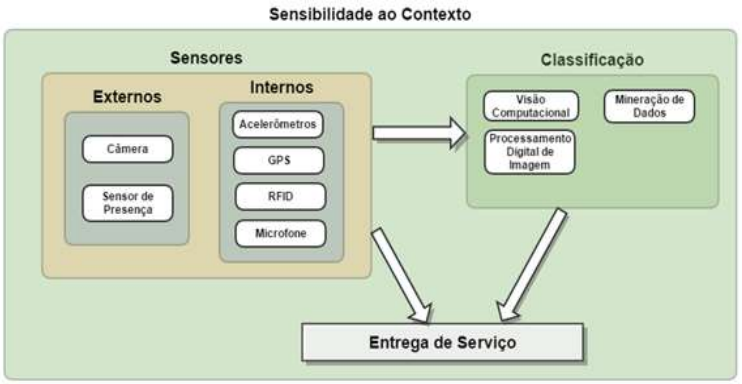

Fig. 1. Diagrama de Sensibilidade ao Contexto

Como se pode perceber pela contextualização abordada na Figura 1, HAR se insere nas aplicações de sensibilidade ao contexto, que podem ser bem complexas e vão desde a coleta de dados de sensores diversos até a entrega de um serviço relevante, preciso e dinâmico. Isto tem alto grau de dependência dos dispositivos instalados, da capacidade de embarcar e dos requisitos da aplicação. O foco aqui dado é no desenvolvimento de classificadores aplicados a um conjunto de dados públicos referentes a acelerômetros tri-axiais.

Este trabalho está organizado da seguinte forma: na Seção II descreve em detalhes os objetivos deste trabalho. Na Seção III a metodologia é apresentada. Na Seção IV são apresentados os resultados e análises dos testes computacionais. Na última Seção apresentam-se as conclusões.

II. OBJETIVOS

O objetivo é implementar um programa Java, com a utilização da biblioteca *Open Source* Weka, biblioteca de código aberto chamada "*Waikato Environment for Knowledge Analysis*" [5] para a aplicação de métodos de classificação, com treinamento e testes, num conjunto de dados público (chamado a partir de agora de *dataset*) referente a acelerômetros tri-axiais e dados específicos de voluntários [6].

Os objetivos são listados a seguir:

1. Expandir as aplicações com o *dataset* do trabalho dos autores [6], ao usar mais classificadores com diferentes configurações. Parte-se da premissa de que não existe um único classificador que seja o melhor para todos os tipos de conjunto de dados, como apontado por [4]. Portanto é totalmente útil e esperado aplicar mais de uma técnica de classificação num determinado problema.
2. Comparar técnicas de validação.
3. Verificar as diferenças de treinamento de classificador considerando coletas de voluntários distintos (diferentes características físicas).
4. Avaliar o desempenho dos classificadores com diferentes técnicas de validação.

As características e métodos da abordagem são:

- Rede Neural (Perceptron Multicamadas) e Rede Bayesiana.
- Treinamento Supervisionado.
- Testes para avaliação do classificador.
- Comparação de resultados.

III. METODOLOGIA

A. Metodologia HAR

Conforme ilustrado em [7], é construído um Classificador, exemplificando-se dados de Reconhecimento de Atividade Humana, tais como o sinal coletado de acelerômetro devidamente posicionado quando a pessoa faz o movimento de beber do copo ou virar a página de um livro. As etapas de construção de um Classificador são:

1. Treinamento:

   a. Buscam-se os relacionamentos entre os dados das instâncias de entrada e as classes alvo, utilizando-se dados separados para treinamento.

      i. OBS.: Os dados para treinamento podem ser separados utilizando-se diversas técnicas, como visto mais adiante.

   b. O Classificador é construído baseado num conjunto de dados cujas classificações já são conhecidas – Aprendizado Supervisionado.

2. Testes de Validação.

   a. Com o Modelo (conjunto de regras) construído, ou seja, o Classificador pronto, usa-se outro conjunto de dados (o de Testes).

   b. Com essa aplicação, comparam-se os resultados (a classe à qual pertence cada instância do conjunto de dados) com as classes desejadas.

3. Resultados:

   a. Avaliam-se as taxas de acerto gerais, as Matrizes de Confusão, o Desempenho de Treinamento e construção do modelo e dos Testes de validação.

B. Metodologia de Coleta de Dados

Esta seção visa explicar como foi a coleta e os critérios de obtenção de dados conforme trabalho de [6], onde se obteve o *dataset* público. Note-se que os métodos descritos nesta seção não denotam as ações do presente trabalho, mas sim como foram os passos e considerações explicados pelos referidos autores.

Os dados foram coletados de 4 acelerômetros tri-axiais (x, y e z), 2 horas de atividades feitas separadamente para cada um de 4 voluntários (2 homens e 2 mulheres), totalizando 8 horas de atividades coletadas. A partir da coleta, foi feita a amostragem dos sinais dos sensores. Foram no total 165.633 amostras e consideradas 5 classes de atividades, realizadas de forma controlada e num tempo determinado: Sentado, Sentar-se, Em pé, Levantando-se, Andando.

Os acelerômetros foram instalados no corpo dos voluntários nas seguintes posições: Cintura, Coxa esquerda, Tornozelo direito e Braço direito.

Uma das primeiras etapas a serem feitas com os dados coletados é o seu Pré-Processamento, quando se prepara o conjunto de dados coletados na forma bruta. Isto foi feito por [6] no *dataset* disponibilizado publicamente. Esta ação consiste em amostrar o sinal do sensor convenientemente. Um arquivo no formato *Comma-Separated Values* (CSV) foi fornecido contendo todas as amostras coletadas.

C. Metodologia de Classificação

Sobre a Rede Bayesiana como classificador, como em [8], cada atributo e classe é representado por um nó. Para o caso de atributos todos discretos, cada nó da rede possui uma tabela de

probabilidades, que são calculadas durante sua construção. De forma geral, a construção de uma Rede Bayesiana segue os seguintes passos de aprendizado:

1) Assume-se alguma ordem entre os nós, obtendo-se uma lista.
2) Nessa lista de nós, para cada probabilidade de cada nó é verificado se há dependência dos nós anteriores na lista.
3) Para onde há impacto, cria-se um arco do nó anterior para o atual.
    a. Desta forma tem-se uma rede que é um grafo direcional acíclico.
4) Calculam-se as probabilidades de cada nó dado as dos nós pai – Probabilidade a Priori.
    a. Utiliza as informações dos dados de treinamento, onde se conhecem as probabilidades das classes e as verossimilhanças.
5) No nó da classe, calcula as probabilidades com base nas probabilidades a priori nos nós de dependência.
6) Verifica a acurácia com base nas instâncias de treinamento.
7) Repete passo 1.
    a. Essa iteração é repetida tantas vezes quantas forem necessárias para o algoritmo de treinamento decidir pela melhor rede.

O processo de construção e treinamento da Rede Bayesiana é feito através de uma heurística a ser escolhida. Na biblioteca Weka estão disponíveis, na forma de algoritmo de aprendizado ("*searchAlgorithm*"), conforme visto em [8].

As Redes Neurais Artificiais (RNA), segundo [9], é um modelo matemático ou computacional que tenta simular a estrutura e os aspectos funcionais de uma rede neural biológica. Consiste num grupo interconectado de neurônios artificiais. Isso provê uma poderosa ferramenta de aprendizado, bastante utilizado em Mineração de Dados devido a ter uma boa resposta preditiva. A arquitetura Perceptron Multicamadas (PMC) é a mais famosa RNA, nela os neurônios são agrupados em camadas (uma camada de entrada, uma de saída e uma ou mais camadas escondidas), e existem apenas conexões para frente (*feedforward*). Ela mapeia um conjunto de dados de entrada (instância) num valor de saída apropriado (classe). Utiliza três ou mais camadas de neurônios (nós), com função de ativação não linear, portanto é capaz de distinguir dados separados não linearmente. Cada nó de uma camada se conecta com todos os nós da camada seguinte sob um determinado peso. O algoritmo *backpropagation* de aprendizado supervisionado de rede é o mais comumente utilizado em diversas aplicações.

Um diagrama esclarecedor da metodologia e passos utilizados são mostrados em [7]. Em um primeiro momento foram feitos testes utilizando a própria Interface Gráfica de Usuário Weka (Weka GUI), disponibilizada juntamente com a biblioteca Java. Ela facilita a visualização dos dados carregados e das funcionalidades disponíveis na biblioteca *Open Source*.

Um programa Java que incorpora a biblioteca *Open Source* Weka 3.6.12 foi implementado tendo em vista os objetivos referentes à metodologia de avaliação dos classificadores construídos.

Uma das desvantagens da utilização somente da interface gráfica Weka GUI é a falta de um maior controle de como os dados são carregados. Quando se carregam os dados do arquivo em formato ARFF as opções de técnicas de validação *holdout* ou k-*fold cross-validation* não levam em consideração a ordem em que as instâncias se apresentam no arquivo. Por exemplo, se o arquivo tem suas instâncias ordenadas por nome da pessoa, e para cada pessoa for ordenado por classe, considerando a escolha como *holdout* com 66% para treinamento, o Weka GUI irá simplesmente cortar as 34% últimas instâncias do arquivo, podendo, assim, ser desconsiderada erroneamente parte dos dados de uma pessoa e/ou os dados completos de uma pessoa. Esta separação de pessoa remete a uma técnica de validação diferente, chamada *leave-one-person-out*. O correto para *holdout*, considerando-se independência dos dados e visando maior generalização quanto possível, seria separar 66% para treinamento para cada pessoa e cada classe. Assim, uma implementação mais sofisticada foi realizada para melhor tratamento dos dados a serem utilizados.

Uma outra importante motivação para o desenvolvimento de um programa Java e não somente fazer testes com a interface Weka GUI é o fato que diversas vezes a interface gráfica não faz uso inteligente e otimizado da memória. Por diversas vezes, mesmo com uma configuração de algoritmo de construção e treinamento da rede simplificado (com restritos limites para espaço de busca pela rede), obteve-se o erro de insuficiência de memória

Foi utilizado o Java versão JDK 1.7.0_71 para Windows, incorporando a biblioteca *Open Source* Weka 3.6.12. A plataforma para o desenvolvimento e testes foi: Memória RAM de 8 GB, Processador Intel Core i7 2.20 GHz, Windows 7, 64 bits, Eclipse Java EE IDE for Web Developers. Version: Kepler Service Release 1.

IV. RESULTADOS

Foram feitas duas abordagens para carga de dados, pré-processamento, construção e treinamento da rede, e testes de validação para obtenção da taxa de acerto e Matriz de Confusão: Weka GUI e Programa Java.

A interface gráfica Weka GUI permite uma melhor visualização dos dados quando carregados, permite agilidade para modificar os parâmetros de configuração dos classificadores.

O programa Java desenvolvido, utilizando a biblioteca Weka, permitiu uma maior clareza e flexibilidade para alterações necessárias de configuração, melhor monitoramento do tempo gasto em construção, treinamento e testes dos classificadores e maior flexibilidade na carga dos dados, os

quais puderam ter suas características devidamente consideradas e tratadas no programa.

A Rede Neural (Perceptron Multicamadas) apresentou bons resultados em todos os testes com ela, tanto com Weka GUI como no programa Java.

Uma das Redes Bayesianas mostrou o melhor resultado de todos, ao ser testada com o programa Java, para o uso da técnica 10-*fold cross validation* para separação de dados de teste e treinamento, *Hill Climber* como algoritmo de aprendizado para construção da rede, a qual teve 2 nós como limite de nós pai por cada nó da rede.

A Rede *Naive Bayes* testada com o Weka GUI apresentou o pior resultado de todos.

A. Resultados com Weka GUI

Em relação aos dados obtidos com o uso da interface gráfica Weka GUI, um resumo pode ser visto nas Tabelas I, II e III. Detalhes referentes às Matrizes de Confusão de cada teste são encontrados em [7]. Os valores "Acerto %" refletem o acerto percentual de uma execução de cada teste. Foram utilizadas duas técnicas de seleção de atributos: no teste'WB2' foi o Algoritmo Genético (parametrização: probabilidade de crossover com valor 0,6, número máximo de gerações aceito igual a 20, probabilidade de mutação como 0,033), reduzindo os atributos de 18 para 10 (apenas alguns atributos relacionados aos acelerômetros foram selecionados), e no teste 'WB3' foi feita uma seleção manual (considerados apenas os dados dos acelerômetros, restando 13 atributos, ignorando os atributos físicos dos indivíduos).

TABELA I. RESULTADOS COM WEKA GUI – REDE BAYESIANA. TESTES 1 A 3.

Teste Nº	WB1	WB2	WB3
Rede	Bayesiana	Bayesiana	Bayesiana
Validação	10-*fold cross-validation*	*holdout* com 66% para treinamento	*holdout* com 66% para treinamento
Algoritmo de Pesquisa	*Naive Bayes*	*Hill Climber*	*Hill Climber*
Algoritmo de Estimação	*Simple Estimator*	*Simple Estimator*	*Simple Estimator*
Seleção de Atributos	Não	Sim (Algoritmo Genético)	Sim
Observações	N/A	Seleção de Atributos: Redução 18 para 10 atributos, sendo todos de acelerômetros.	Seleção de Atributos: Apenas os dados de todos os 4 acelerômetros.
Acerto %	91,0651	89,5676	91,0326

TABELA II. RESULTADOS COM WEKA GUI – REDE BAYESIANA. TESTES 4 A 6.

Teste Nº	WB4	WB5	WB6
Rede	Bayesiana	Naive Bayes	Bayesiana
Validação	*holdout* com 66% para treinamento	*holdout* com 66% para treinamento	*leave-one-person-out* (katia)
Algoritmo de Pesquisa	*Hill Climber* (global, com ajustes de configuração)	N/A	K2 (ajustes de configuração)
Algoritmo de Estimação	*Simple Estimator*	N/A	*Simple Estimator*
Seleção de Atributos	Não	Não	Não
Observações	N/A	N/A	O arquivo de dados foi manualmente dividido em 2: Testes, só com "katia". Treinamento, com os outros.
Acerto %	91,5919	77,0505	54,2021

TABELA III. RESULTADOS COM WEKA GUI – REDE NEURAL (PERCEPTRON MULTICAMADAS).

Teste Nº	WN1
Rede	Rede Neural (Perceptron Multicamadas)
Validação	*holdout* com 66% para treinamento
Épocas	500
Taxa de	0,3
Momentum	0,2
Acerto %	97,4234

Nestes testes com Weka GUI, a Rede Neural (Perceptron Multicamadas) apresentou o melhor resultado, e a Rede *Naive Bayes* apresentou o pior resultado. Detalhes referentes às Matrizes de Confusão de cada teste são encontrados em [7].

B. Resultados com Programa Java

Em relação aos dados obtidos com o programa Java, um resumo pode ser encontrado nas Tabelas IV, V e VI. Os valores "Acerto %" refletem o acerto percentual de uma execução de cada teste.

TABELA IV. RESULTADOS COM PROGRAMA JAVA – REDE BAYESIANA. TESTES 1 A 3.

Teste Nº		JB1	JB2	JB3
Rede		Bayesiana	Bayesiana	Bayesiana
Validação		leave-one-person-out (katia)	*holdout* com 66% para treinamento	*holdout* com 66% para treinamento
Algoritmo de Pesquisa		*Hill Climber*	*Hill Climber*	*Hill Climber*
Algoritmo de Estimação		*Simple Estimator*	*Simple Estimator*	*Simple Estimator*
Número máx Nós Pai		1	1	1
Performance	**Construção e Treinamento (s)**	46, 426	56,589	55,861
	Testes (s)	0,424	0,29	0,283
Acerto %		30,0098	86,8561	88,7188

TABELA V. RESULTADOS COM PROGRAMA JAVA – REDE BAYESIANA. TESTES 4 A 7.

Teste Nº		JB4	JB5	JB6	JB7
Rede		Bayesiana	Bayesiana	Bayesiana	Bayesiana
Validação		leave-one-person-out (katia)	leave-one-person-out (wallace)	*holdout* com 66% para treinamento	*10-fold cross validation*
Algoritmo de Pesquisa		*Hill Climber*	*Hill Climber*	*Hill Climber*	*Hill Climber*
Algoritmo de Estimação		*Simple Estimator*	*Simple Estimator*	*Simple Estimator*	*Simple Estimator*
Número máx Nós Pai		2	2	2	2
Performance	Construção e Treinamento(s)	4787,289	4129,986	4322,115	5540,887
	Testes (s)	0,416	0,336	0,246	51458,738
Acerto %		34,9921	32,4944	97,4544	97,9282

TABELA VI. RESULTADOS COM PROGRAMA JAVA – REDE NEURAL (PERCEPTRON MULTICAMADAS).

Teste Nº		JN1	JN2
Rede		Rede Neural (Perceptron Multicamadas)	Rede Neural (Perceptron Multicamadas)
Validação		*holdout* com 66% para treinamento	*holdout* com 66% para treinamento
Épocas		1000	5000
Taxa de Aprendizado		0,3	0,3
Momentum		0,2	0,6
Performance	Construção e Treinamento(s)	781,849	3894,841
	Testes (s)	782,084	3895,117
Acerto %		95,5106	96,4380

Percebemos que a Rede Bayesiana mostrou o melhor resultado (97,9282% de acerto) após ser construída admitindo-se até 2 nós pai por nó da rede, adotando-se a técnica de validação 10-*fold cross validation*, tomando ao mesmo tempo o maior tempo para construir e treinar (5540,887 segundos) e testar (51458,738 segundos), ou seja, teve o menor desempenho no tempo computacional.

O pior resultado foi obtido com Rede Bayesiana ao se utilizar a técnica de validação *leave-one-person-out*, com a "katia", com no máximo 1 nó pai por nó da rede (30,0098% de acerto). Mesmo ao trocar a pessoa ("wallace") a ser separada para teste de validação, o resultado continua muito ruim, em torno de 30% de taxa de acerto.

Os resultados para Rede Neural (Perceptron Multicamadas) foram muito bons, tendo 95,5106% e 96,438% de taxa de acerto nas duas tentativas, após pequenas alterações nas configurações de número de épocas e de *momentum*. Uma das Redes Bayesiana mostrou o melhor resultado de todos, 97,9282%, para o uso da técnica 10-*fold cross validation* para separação de dados de teste e treinamento, *Hill Climber* como algoritmo de aprendizado para construção da rede, a qual teve 2 nós como limite de nós pai por cada nó da rede. Em ambos os testes, o número máximo de épocas foi atingido.

Ao se testar a Rede Bayesiana com maior número máximo de nós pai para cada nó da rede, por exemplo 3 (o que seria uma rede denominada *Bayes Net Augmented Bayes Network*, BAN), ou até mesmo 1000 (o que configura uma quantidade ilimitada para a biblioteca Weka), os resultados teriam potencial para melhorar um pouco, entretanto devido às limitações da plataforma de testes houve estouro de memória. Notou-se um uso constante de pelo menos 5 Giga bytes de memória RAM, chegando rapidamente a um estouro (obs.: o computador de teste tem no máximo 8 Giga bytes de memória RAM). Notamos que houve tentativas de configurar um maior espaço de pilha de memória para o programa, chegando-se a 6 GB num computador de 8 GB, mas mesmo assim resultou em estouro de pilha.

V. CONCLUSÃO

De acordo com os resultados apresentados na Seção IV, podemos fazer conclusões com base em comparações entre eles e nas metas deste trabalho.

Tanto com a Rede Neural (Perceptron Multicamadas) como com a Rede Bayesiana obtiveram-se excelentes resultados, com mais de 97% de taxa de acerto. A Rede Neural possui o melhor desempenho (tempo de processamento) na construção e treinamento e nos testes, obtendo simultaneamente excelentes taxas de acerto. A Rede Bayesiana atingiu excelentes taxas de acerto quando se utilizou a técnica de validação 10-*fold cross-validation*, e configurando-se o número máximo de nós pai para cada nó da rede com valor pelo menos 2. Contudo tanto a construção da rede como os testes de validação da Rede Bayesiana são extremamente custosos (elevado tempo, por exemplo 14 horas para testes de validação, enquanto a Rede Neural chegou a levar aproximadamente 1 hora nos testes validação).

Na Rede Bayesiana, ao se aplicar diferentes técnicas de Validação (*holdout* e 10-*fold cross-validation*), com e sem seleção de atributos, obtiveram-se resultados em torno de 90% para a maioria. A técnica de validação 10-*fold cross-validation* apresentou o melhor resultado global de todos os testes (97,9282%). A técnica de validação *holdout* é bastante eficaz para o problema proposto, principalmente quando se leva em consideração as características e ordenação das instâncias no arquivo de dados carregado (tal como foi feito no Programa Java). A técnica de validação *leave-one-person-out* se mostrou bastante ruim, com resultados de percentual de acerto abaixo de 35% em todas as tentativas, independentemente do número máximo de nós pai ou de outras configurações do algoritmo de construção da Rede Bayesiana (*Hill Climber* ou *K2*). Isto revela que o treinamento, da forma que é realizado, considerando-se a metodologia de coleta de dados, não é útil para ser expandido para pessoas diferentes, ou seja, com diferentes comportamentos físicos, que impactem as medições dos acelerômetros. Uma abordagem mais avançada com essa técnica, considerando por exemplo maior quantidade de indivíduos para coleta de dados, seria relevante em trabalhos futuros.

Selecionar atributos na Rede Bayesiana, na tentativa de remover os não relevantes, não causou impacto significativo neste problema, mesmo ao se aplicar métodos sofisticados de seleção de atributos, tal como o algoritmo Genético.

Mudanças na técnica heurística de construção da Rede Bayesiana se mostrou promissor. Obteve-se o melhor resultado dos testes ao mudar de *Hill Climber* local para *Hill Climber* global. Entretanto, ao se tentar mudar de *Hill Climber* para *K2*, que é semelhante mas atrelado à ordem dos nós da rede, há grande piora do resultado (chegou-se a uma taxa de acerto de aproximadamente 54%).

A Rede *Naive Bayes* como classificador (também chamado de Classificador Ingênuo), a qual é bem mais simples pois assume que terá apenas o nó da classe ligada com um arco a todos os outros nós dos atributos, se mostrou bastante ineficaz para o problema proposto. Sua baixa taxa de acerto revela que não existe completa independência entre os atributos. Isso já era de certa forma esperado tendo em vista que os dados dos usuários dependem, por exemplo, da idade, peso e altura, pois eles impactam na forma como os movimentos são realizados e nos valores coletados dos acelerômetros. Sabe-se que atributos correlacionados podem causar degeneração em relação ao desempenho do Classificador Ingênuo porque a condição de independência condicional não vale mais para aqueles atributos.

No trabalho de [6], onde se publicou o *dataset* aqui utilizado, obteve-se uma taxa de acerto de 99,4%. Os autores utilizaram o classificador *AdaBoost*, o qual combina árvores de decisão, e técnica de validação 10-*fold cross validation*. Portanto, comparando-se as taxas de acerto, este trabalho apresenta um bom resultado, isto é, 97,9282%. Note-se que no trabalho de [6] não há detalhamento da plataforma computacional utilizada nem dos resultados de desempenho (tempo de construção, treinamento e validação do classificador). Neste trabalho, o melhor resultado encontrado (97,9282%) foi para Rede Bayesiana, com algoritmo de aprendizado *Hill Climber*, número máximo de nós pai igual a 2, com 10-*fold cross validation*. Sendo assim, considerando-se os tempos gastos para construção, treinamento e testes encontrados, concluímos que os resultados do presente trabalho são satisfatórios.

AGRADECIMENTOS

Os autores agradecem as sugestões dos revisores anônimos com suas recomendações nos quais enriqueceram o trabalho.

REFERÊNCIAS

[1] W. Ugulino, M. Ferreira, E. Velloso e H. Fuks. Virtual Caregiver: Colaboração de Parentes no Acompanhamento de Idosos. Anais do SBSC 2012, IX Simpósio Brasileiro de Sistemas Colaborativos, pp. 43-48. São Paulo, SP: IEEE. ISBN 978-0-7695-4890-6, 2012.

[2] H. Gjoreski, M. Lustrek e M. Gams. Accelerometer Placement for Posture Recognition and Fall Detection. In: 7th International Conference on Intelligent Environments, IE, 2011.

[3] J. Lockhart. Mobile Sensor Data Mining, Fordham Undergraduate Research Journal, vol. 1, pp 67-68, 2011.

[4] G. Kesavaraj e S. Sukumaran. A study on classification techniques in data mining, Computing, Communications and Networking Technologies (ICC-CNT),2013 Fourth International Conference on pp. 4-6, 2013.

[5] Weka 3: Data Mining Software in Java. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka>. Acesso em: 03 nov. 2014.

[6] W. Ugulino, D. Cardador, K. Vega, E. Velloso, R. Milidiu e H. Fuks. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science, pp. 52-61. Curitiba, PR: Springer Berlin/Heidelberg. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6, 2012.

[7] L. M. Rodrigues. Métodos de Classificação para Reconhecimento de Atividade Humana. 2015. 132 f. TCCP (Especialização em Engenharia Elétrica com Ênfase em Sistemas Inteligentes Aplicados à Automação) – Instituto Federal do Espírito Santo, Vitória, 2015 Disponível em: <https://biblioteca2.ifes.edu.br/vinculos/00000C/00000CF5.pdf>. Acesso em : 19 maio 2015.

[8] R. R. Bouckaert. Bayesian Network Classifiers in Weka for Version 3-5-7. The University of Waikato, 2008.

[9] A. Rakhman, G. Y. Sun e R. Catur. Building Artificial Neural Network Using WEKA Software. Information System Department, Sepuluh Nopember Institute of Technology at Surabaya, Indonesia, 2009.