Avaliação de técnicas de inteligência computacional para identificação de atividades de vida diária

Wylken S. Machado[1], **Pedro H. Barros**[1], **Eliana S. Almeida**[1], **Andre L. L. Aquino**[1]

[1]Instituto de Computação, Universidade Federal de Alagoas, Maceió, Alagoas, Brasil.
wylken.ufal@gmail.com,pedro_h_nr@laccan.ufal.br
{eliana.almeida,alla}@laccan.ufal.br

Abstract. *This work presents the performance evaluation of machine learning algorithms to identify Activity Daily Living (ADLs) and falls.We assess the following algorithms: K-Nearest Neighbors, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, Extra-Trees and Recurrent Neural Network. We use a dataset collected through a Body Sensor Network with five sensor devices connected through Bluetooth Low Energy interface, called UMAFall. Satisfactory results were found, mainly for ADL sitting down and frontal fall, with an accuracy of* $100\,\%$ *with Extra-Trees algorithm.*

Resumo. *Neste trabalho apresentamos a avaliação do desempenho de algoritmos de machine learning para identificar Atividades de Vida Diária (ADLs) e quedas.Nós avaliamos os seguintes algoritmos: K-Nearest Neighbors, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, Extra-Trees e Redes Neurais Recorrentes. Utilizamos um conjunto de dados coletados por uma Body Sensor Networks com cinco dispositivos sensores conectados através da interface Bluetooth Low Energy, chamado UMAFall. Obtivemos resultados satisfatórios, principalmente para as atividades saltar e queda frontal, com* $100\,\%$ *de acurácia, utilizando o algoritmo Extra-Trees.*

1. Introdução

O monitoramento da vida diária pode ser utilizado para ajudar idosos ou pessoas com doenças crônicas, promovendo melhorias na qualidade de vida. Quando tecnologias como Smart Homes são utilizadas para ajudar pessoas com necessidades especiais, temos o conceito de Health Smart Home (HSM), que emerge da combinação da telemedicina, automação residencial e sistemas da informação, definida como uma casa inteligente com dispositivos de assistência a saúde e que podem ser acessados remotamente [Mano et al. 2016].

Com o intuito de melhorar aplicações de inferência sobre a estrutura de uma HSM, este trabalho apresenta uma avaliação dos principais algoritmos de aprendizado de máquina para identificar atividades da vida diária (ADLs) e quedas [Katz 1984]. Para isso, um conjunto de dados gerados por uma Body Sensor Networks (BSN), chamada UMAFall, foi utilizado [Casilaria et al. 2017b].

Esta base de dados possui cinco dispositivos sensores, conectados por Bluetooth, para identificar atividades diárias de dezessete indivíduos. São elas: caminhar, correr, inclinar o corpo, pular, deitar, sentar, queda lateral, queda frontal e queda para trás. Cada dispositivo grava simultaneamente os valores do acelerômetro, giroscópio e

magnetômetro. Sobre esses dados foram aplicados seis algoritmos de aprendizado de máquina:K-Nearest Neighbors (k-NN) [Shalev-Shwartz and Ben-David 2014], Naive Bayes [Smola and Vishwanathan 2008], Support Vector Machine (SVM) [Micucci et al. 1995], Decision Tree [Shalev-Shwartz and Ben-David 2014], Random Forest [Breiman 2001], Extra Trees [Geurts et al. 2006] e Redes Neurais Recorrentes [Goodfellow et al. 2016]. Para cada algoritmo foi verificado o desempenho alcançado e o tempo de execução na detecção de ADLs e quedas. Também foi verificado qual dos sensores espalhados pelo corpo apresentou o melhor resultado.

A principal contribuição deste artigo é a análise da acurácia alcançada pelos algoritmos na detecção de ADLs e quedas utilizando o conjunto de dados público UMAFall, além de verificar o tempo de execução e qual a melhor parte do corpo para utilizar sensor que irá coletar as informações dos movimentos, contribuindo com futuros trabalhos que desejam realizar a detecção de ADLs e quedas. Essa análise consiste no estudo dos seis algoritmos citados anteriormente para identificar qual é o mais apropriado para aplicações que utilizam BSN, com as mesmas características do UMAFall, para detectar atividades diárias , e qual a melhor parte do corpo para se instalar os sensores (tórax, pulso, tornozelo, cintura e bolso).

2. Trabalhos Relacionados

Existem vários estudos relacionados ao monitoramento de atividades das pessoas em suas residências. Existem alguns trabalhos na literatura, como proposto por Mano et al. [Mano et al. 2016], que utiliza imagens de câmeras para ajudar enfermeiras e cuidadores a identificar emoções nos pacientes, ajudando no processo de recuperação. Outro trabalho, proposto por Aloulou et al. [Aloulou et al. 2013], realiza a implantação de um sistema HSM em um ambiente real para avaliar seu desenvolvimento e usabilidade. Este sistema consiste de sensores e dispositivos controlados por software para detectar precocemente a degradação das condições do paciente. Trabalhos como o proposto por Gope et al. [Gope and Hwang 2016], já abordam o conceito de Body Sensor Networks (BSNs) e os problemas relacionados à segurança da informação que essas aplicações podem apresentar.

Outros trabalhos abordam os métodos de reconhecimento de Atividade da Vida Diária (ADLs). Khan et al. [Khan et al. 2010] utiliza os dados gerados pelo acelerômetro de um smartphone para detectar quatro ADLs, alcançando uma acurácia de até $96\,\%$, apesar de algumas limitações quanto ao posicionamento e orientação do smartphone. Mohammad et al. [Alam and Roy 2017] propõe utilizar dados gerados por uma BSN contextualizada e técnicas de deconvolução para o reconhecimento de ADLs. Farhad et al. [Shahmohammadi et al. 2017] propõe a utilização de dados gerados pelo acelerômetro e giroscópio de um smartwatch para calcular nove características estatísticas e, utilizando essas características, identificar cinco ADLs. Seus resultados apresentaram uma acurácia de até $92\,\%$. Weiss et al. [Weiss et al. 2016] faz uma comparação entre a acurácia alcançada na identificação de 18 ADLs, utilizando dados gerados por um smartphone e um smartwatch, concluindo que os dados gerados pelo smartwatch apresentaram melhores resultados, principalmente para ADLs que são executadas predominantemente com as mãos.

Existe uma variedade de técnicas e soluções para o problema apresentado. Con-

tudo, é necessário um estudo comparativo mais aprofundado sobre os impactos das técnicas de Machine Learning no cenário de detecção de ADLs, utilizando diferentes tipos de sensores em várias partes do corpo, e assim, contribuir para a escolha do melhor local de instalação dos sensores. Além disso, o trabalho apresenta o tempo de execução dos algoritmos para classificar as ADLs.

3. Algoritmos de Machine Learning

Nessa seção é apresentada uma breve descrição dos algoritmos de Machine Learning utilizados neste estudo. Todos utilizam a abordagem de aprendizado supervisionado. Como esse trabalho se propõe a verificar a acurácia de diferentes técnicas, os algoritmos foram escolhidos com o objetivo de representar os principais métodos matemáticos e computacionais aplicados ao Aprendizado de Máquina.Foi utilizada o tipo de rede neural RNN para possibilitar a captura de features temporais.

O primeiro algoritmo utilizado foi o k-Nearest Neighbors (k-NN) [Cover and Hart 1967] é um algoritmo de aprendizado supervisionado baseado em instâncias, onde é necessário ter uma base de dados rotulada para realizar o treinamento do classificador e identificar novos elementos. Essencialmente o k-NN calcula as distâncias entre um elemento desconhecido e os elementos do conjunto de treinamento. Existem vários métodos para calcular a distância, sendo que a mais utilizada é a distância Euclidiana [Mulak and Talhar 2015],

$$d(x, x') = \sqrt[2]{(x_1 - x'_1)^2 + (x_2 - x'_2) + ... + (x_n - x_n)'}. \quad (1)$$

Além da distância Euclidiana, também são utilizadas as distâncias de *Chebyshev* e *Manhattan*. O algoritmo consiste em percorrer todo o conjunto e computar a distância d entre x e as observações de treinamento. Logo após, escolhe os k elementos mais próximos a x formando o conjunto A. Então, estima-se a probabilidade condicional para cada classe em A

$$p(y = j|X = x) = \frac{1}{k} \sum_{i \in A} I(y_i = j), \quad (2)$$

onde $I(z)$ retorna **1** quando o argumento z for verdadeiro e **0** se for falso. E, por fim, atribui-se a classe com maior probabilidade a x.

Um passo essencial para os cálculos do k-NN é definir o valo de k, que deve ser um valor ímpar para evitar complicações na inferência. Além disso, se o valor de k for muito pequeno, a classificação pode ser sensível a ruído, e se for muito grande, pode incluir elementos de outras classes. Nesse trabalho o valor que apresentou o melhor resultado foi $k = 5$.

O segundo algoritmo considerado foi o Naive Bayes. Amplamente discutido em [Lewis 1998], este método utiliza uma abordagem probabilística baseada no Teorema de Bayes,

$$p(c|x) = \frac{p(x|c)p(c)}{p(c)}. \quad (3)$$

Os atributos são considerados condicionalmente independentes, dado uma classe C_k. Ou seja, a presença de uma característica não está relacionado a nenhuma outra.

Dado um elemento $x = \{x_1, x_2, ..., x_n\}$ com n atributos, o Naive Bayes prediz a classe C_k para x utilizando a probabilidade calculada pelo Teorema de Bayes,

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{p(x_1, x_2, ..., x_n|C_k)p(C_k)}{p(x_1, x_2, ..., x_n)}, \tag{4}$$

onde C é o conjunto com as k classes possíveis. Após calcular as probabilidade é atribuído a classe C_k com maior probabilidade ao elemento x.

O terceiro algoritmo considerado foi o Support Vector Machine (SVM).No nosso caso O objetivo do algoritmo é encontrar o hiperplano que melhor separa as classes do conjunto de treinamento, ou seja, que maximiza sua margem entre as classes do conjunto.

Os elementos mais próximos do hiperplano são chamados de vetores de suporte. Encontrar esses vetores é a principal tarefa da fase de treinamento do SVM. A classificação dos novos elementos é realizada utilizando apenas os vetores de suporte. Após o treinamento o SVM pode descartar todo o resto dos dados de treinamento, o que faz com que esse algoritmo seja bastante eficiente para realizar a classificação, já que utiliza apenas alguns vetores de suporte, e não todo o conjunto.

Em conjuntos não separáveis linearmente, o SVM utiliza funções matemáticas chamadas de kernels que transformam esse conjunto em outro linearmente separável. Normalmente são utilizados os kernels: linear, polinomial, radial e sigmoide. Para construir um hiperplano ótimo, o SVM aplica um algoritmo de treinamento iterativo, utilizado para maximizar a margem entre esse hiperplano e os elementos de diferentes classes.

Suponha que temos um conjunto de dados $[(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)]$ com classes linearmente separáveis, onde $y_i = -1$ para x_i da Classe 0 e $y_i = 1$ para x_i da Classe 1. A reta que separa essas duas classes pode ser definida como $\overrightarrow{w} \cdot \overrightarrow{x} + b = 0$, onde $\overrightarrow{w}$ e $\overrightarrow{x}$ são dois vetores dimensionais. Definimos o vetor negativo $\overrightarrow{x_n}$ em que todas as entradas são da Classe 0, e o vetor positivo $\overrightarrow{x_p}$ onde todas as entradas são da Classe 1. Agora podemos definir os limites inferior e superior da nossa reta: $\overrightarrow{w} \cdot \overrightarrow{x_n} + b = -1$ para Classe 0, e $\overrightarrow{w} \cdot \overrightarrow{x_p} + b = 1$ para Classe 1. A distância entre esses dois limites é dada por $\frac{2}{||\overrightarrow{w}||}$, e a margem M é dada por $\frac{1}{||\overrightarrow{w}||}$. Para maximizar M devemos minimizar $||\overrightarrow{w}||$. O SVM utilizara técnicas para resolver problemas de otimização com o objetivo de minimizar $||\overrightarrow{w}||$.

Esse é um exemplo em um ambiente com duas dimensões. A mesma técnica pode ser utilizada para N dimensões.

Em seguida avaliamos os algoritmos de Árvore de Decisão [Quinlan 1986].Segundo Shalev-Shwartz et al. [Shalev-Shwartz and Ben-David 2014], uma Árvore de Decisão é um preditor $h : x \rightarrow y$ que prediz a classe associada com a uma instância x percorrendo uma estrutura de dados em árvore, dos seus nós raízes até um nó folha. Uma Árvore de Decisão é composta por vários nós. As folhas contêm as classes de interesse, e os nós intermediários representam as características que definem as classes. Esse classificador é de fácil entendimento e interpretação, sendo essa uma das vantagens em utilizá-lo.

Para obter bons resultados se faz necessário escolher métricas adequadas para selecionar as características que irão formar os nós. Geralmente é utilizado o Ganho de

Informação que usa o conceito de entropia,

$$h(x) \quad = \quad \sum_{i=1}^{k} p_i \log_2(p_i), \tag{5}$$

onde x é o atributo, k é a quantidade de classes do conjunto de dados e p_i é a probabilidade de cada uma das classes acontecer para um dado atributo. Para o cálculo do Ganho de Informação utilizamos

$$ig(x) = h(classe) - h(x). \tag{6}$$

Uma variação da árvore de decisão é o algoritmo Random Forest [Breiman 2001] que cria diversas árvores de decisão e, para cada árvore, k atributos são escolhidos aleatoriamente para formá-la. Após a escolha dos atributos pelo sistema, as árvores são construídas da forma tradicional, utilizando o ganho de informação. Sua estrutura permite executar o algoritmo de forma distribuída, tornando-se eficiente para problemas de classificação em enormes bases de dados. A acurácia tende a aumentar de acordo com o número de árvores utilizadas.

O algoritmo Extra-Trees [Geurts et al. 2006] é similar ao Random Forest. A diferença entre eles é que o Extra-Trees adiciona mais uma camada de aleatoriedade para montar as árvores, ou seja, o algoritmo utiliza uma estratégia aleatória para montar os nós, em vez do ganho de informação ou outras métricas.

Por fim, consideramos o algoritmo baseado em redes neurais recorrentes, ou RNN. Uma Rede neural tradicional (FNN) tem seus parâmetros separados para cada vetor de features de entrada, ou seja, esta necessita aprender todas as regras separadamente. Diferentemente disto, as RNN compartilham os mesmos pesos através do tempo.

Logo, para um modelo dinâmico clássico com dependência temporal, a rede pode ser caracterizada como,

$$s^{(t)} = f(s^{(t-1)}; \theta), \tag{7}$$

onde s^t é um estado do sistema no tempo t e θ é um vetor de features de entrada para o sistema.

Vemos em Goodfellow et al. [Goodfellow et al. 2016] que a RNN representada na Equação 8 é universal (como pode ser visto na equação 7 que representa a mesma equação de um modelo dinâmico clássico) no sentido de que qualquer função que pode ser computada por uma Máquina de Turing pode ser calculada por uma rede recorrente de tamanho finito. Ainda em Goodfellow et al. [Goodfellow et al. 2016], vemos que as RNN são caracterizadas pelas equações:

$$\mathbf{a}^{(t)} = b + W\mathbf{s}^{(t-1)} + U\mathbf{x}^{(t-1)} \tag{8}$$

$$\mathbf{s}^{(t)} = tanh(\mathbf{a}^{(t)}) \tag{9}$$

$$\mathbf{d}^{(t)} = c + V\mathbf{s}^{(t)} \tag{10}$$

$$\mathbf{o}^{(t)} = softmax(\mathbf{d}^{(t)}), \tag{11}$$

onde $\mathbf{x}^{(t-1)}$ é a entrada do passo temporal t, b e c são os vetores de viés das redes neurais, U, V e W são, respectivamente, matriz de peso para as conexões entrada-camada oculta, camada oculta-saída e camada oculta-camada oculta.

Porém, como a RNN tem dificuldades de aprender longas dependências temporais por conta do problema do desaparecimento do gradiente (intrínseco ao treinamento de redes neurais artificiais), é indicado usar um tipo especial de RNN chamada LTSM (Long Short Term Memory), devido a adição de alguns elementos internos de memória. Para este trabalho foi usada a arquitetura proposta por Karien et al. [Karim et al. 2018] onde os autores utilizaram uma rede LTSM concatenada com algumas camadas de redes neurais para refino na performance dos modelos treinados.

4. Base de dados

Foram encontradas várias bases de dados para detecção de ADLs e quedas, tais como [Ojetola et al. 2015], [Vavoulas et al. 2013], [Vavoulas et al. 2016], [Gasparrini et al. 2014], [Sucerquia et al. 2017] e [Micucci et al. 2016]. Porém, a UMAFall[1] [Casilaria et al. 2017a] foi a que apresentou uma maior quandidade de dispositivos e sensores espalhados pelo corpo, característica fundamental para o estudo realizado.

O UMAFall contém o registro de uma BSN que utiliza três tipos de sensores, acelerômetro, giroscópio e magnetômetro. Ela identifica seis tipos de ADLs: caminhar, correr, inclinar o corpo, pular, deitar, sentar, e três tipos de quedas: queda lateral, queda frontal e queda para trás. A BSN utilizada consiste de um smartphone android colocado no bolso direito dos voluntários, usado como nó central, e quatro dispositivos sem fio posicionados em diferentes partes do corpo: tórax, pulso, tornozelo e cintura. Durante a coleta de dados foram utilizados dois modelos de smartphones, Samsung S5 e LG G4, ambos integrados com acelerômetro e conectividade Bluetooth. Cada dispositivo sem fio consiste em um SensorTag CC2650 Multi-Standard da Texas Instruments, que possui interface Bluetooth Low Energy, acelerômetro tri-axial, giroscópio tri-axial e magnetômetro. Cada dispositivo utiliza uma bateria de lítio CR2032 3v.

Originalmente o UMAFall consiste em 531 arquivos CSV, onde cada um é nomeado de forma a identificar o voluntário, a ADL ou o tipo de queda e a data. Dentro de cada arquivo encontramos os dados: timestamp, número da amostra, eixo X, eixo y, eito z, tipo do sensor (acelerômetro, giroscópio e magnetômetro) e local do sensor (bolso, cintura, tornozelo, pulso e tórax).

5. Metodologia

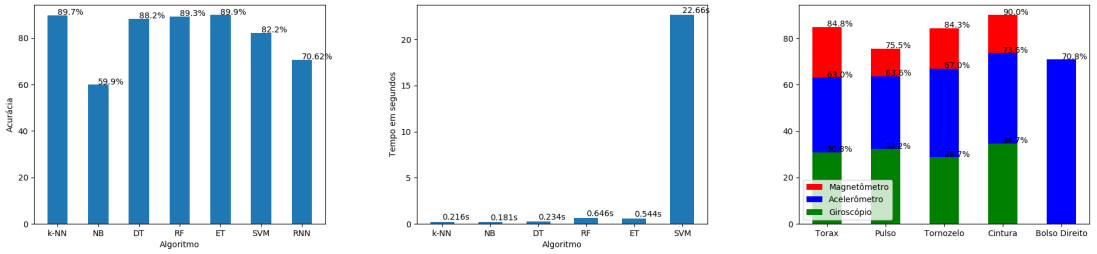
Todo o experimento foi realizado utilizando os dados contidos no conjunto de dados UMAFall com o total de 3.277.950 registros. Foram utilizados os dados individualizados por voluntário, ou seja, os algoritmos executaram a classificação utilizando os dados gerados por uma única pessoa em cada interação até passar por todos os voluntários. As características do movimento variam de um indivíduo para outro. Essa abordagem individualizada apresenta melhores resultados [Weiss et al. 2016], porém não pode ser generalizada, ou seja, utilizar dados de indivíduos diferentes para os conjuntos de treinamento e teste.

[1]Dataset público disponível em `http://webpersonal.uma.es/de/ECASILARI/Fall_ADL_Traces/UMA_FALL_ADL_dataset.html`

Utilizamos a linguagem de programação Python [Python 2017], versão 3.6.4, em conjunto com a biblioteca de machine learning Scikit-learn [Pedregosa et al. 2011], versão 0.19.1, TersorFlow [Brain 2017], versão r1.6, e o ambiente de desenvolvimento PyCharm Community [JetBrains 2018], versão 2018.1. A RNN foi executada em uma GPU GTX 1080 TI, e os demais algoritmos em um desktop com CPU FX 8300, 3.3GHz e 8GB de memória RAM.

Nosso objetivo é verificar a precisão dos algoritmos utilizando os dados de diferentes sensores, localizados em diversas partes do corpo. Para isso, foi utilizado o método validação cruzada k-fold [Stone 1974]. Nesse método o conjunto de dados é dividido em k subconjuntos, logo após, são realizadas k interações de 1 até k, onde o subconjunto k é utilizado para teste e o restante $(k-1)$ para treinamento do classificador. Nesse trabalho utilizamos $k = 10$, valor normalmente utilizado na literatura.

6. Resultados



(a) Acurácia dos algoritmos. (b) Tempo da classificação. (c) Acurácia por local do corpo.

Figura 1: Acurácia e tempo de execução por algoritmo.

A Figura 1(a) mostra a acurácia obtida com todos os métodos propostos. Verifica-se que o Extra-Trees obteve o melhor resultado, seguido do k-NN e do Random Forest, com $89.9\,\%$, $89.7\,\%$ e $89.3\,\%$, respectivamente. Na figura NB = Naive Bayes, DT = Decision Tree, RF = Random Forest, ET = Extra-Trees. Sabendo que o Extra-Trees obteve o melhor resultado, montamos a Figura 1(c) para identificar qual local do corpo apresentou o melhor resultado. Observamos que o sensor que obteve o melhor resultado foi o magnetômetro instalado na cintura do voluntário. Também podemos identificar que o dispositivo do bolso direito registrou apenas os dados do acelerômetro.

Para analisar com mais detalhes como o Extra-Trees realiza a classificação, utilizamos a matriz de confusão apresentada na Figura 2. Nesta matriz observa-se que todas as ADLs e tipos de queda, exceto a ADL correr, obtiveram uma acurácia superior ou igual a $90\,\%$ que, em termos de machine learning, são resultados promissores. O mais notável foram os resultados para a ADL saltar e para a queda frontal, onde todas as ocorrências foram classificadas de forma correta, alcançando $100\,\%$ de precisão nessas atividades.

Tendo em vista que os algoritmos Extra-Trees, k-NN e Random Forest obtiveram resultados próximos, a Figura 1(b) foi utilizada para verificar qual desses algoritmos seria mais indicado para realizar a classificação em aplicações de tempo real. Verificamos que os três realizaram a classificação em menos de $1\,\mathrm{s}$, porém o k-NN foi duas vezes mais rápido que o Extra-Trees. Nessa situação, para uma aplicação específica, é recomendável

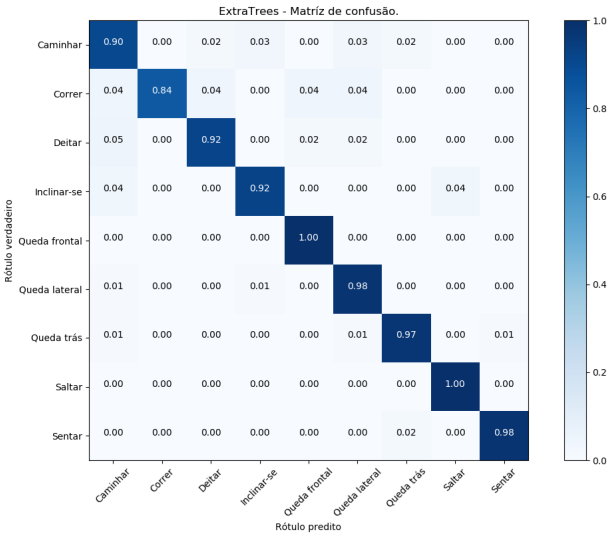


Figura 2: Matriz de confusão utilizando o Extra-Trees com os dados gerados pelo magnetômetro posicionado na cintura.

averiguar qual algoritmo seria o mais indicado, ou seja, se devemos priorizar a acurácia ou o tempo de classificação. O RNN demorou cerca de $27\,\mathrm{min}$ para realizar a classificação, muito mais que os outros algoritmos, dessa forma ele não foi apresentado nesse gráfico.

7. Conclusão

Uma das funções essenciais do monitoramento é a identificação de queda do indivíduo. Uma simples falha nesse tipo de monitoramento pode colocar o indivíduo monitorado em risco. Os resultados apresentam o Extra-Trees, o k-NN e o Random Forest com uma precisão aceitável para esse tipo de aplicação, principalmente para o algoritmo Extra-Trees na identificação da ADL saltar e da queda frontal, com $100\,\%$ de precisão. O Extra-Trees também apresenta uma ótima acurácia para os outros tipos de ADL e quedas. O Extra-Trees e o k-NN apresentaram resultados muito semelhantes, mas o k-NN foi duas vez mais rápido que o Extra-Trees.

Em trabalhos futuros pretendemos utilizar o conhecimento adquirido para elaborar modelos com o intuito de generalizar a classificação, ou seja, utilizar um conjunto de treinamento padrão para realizar detecção de ADLs e quedas de outras pessoas, mantendo a acurácia da abordagem individual. Também desejamos implementar estratégias para descoberta de novas ADLs que não pertençam ao conjunto inicial de treinamento, possibilitando ampliar e personalizar o leque de atividades monitoradas.

Agradecimentos

Nós agradecemos o apoio financeiro ao CNPq, FAPEAL, FAPESP.

Referências

Alam, M. A. U. and Roy, N. (2017). Single bsn-based multi-label activity recognition. *The Third IEEE International Workshop on Sensing Systems and Applications Using Wrist Worn Smart Devices, 2017*.

Aloulou, H., Mokhtari, M., Tiberghien, T., Biswas, J., Phua, C., Lin, J. H. K., and Yap, P. (2013). Deployment of assistive living technology in a nursing home environment: methods and lessons learned. *BMC Medical Informatics and Decision Making*, pages 1–17.

Brain, G. (2017). Tensorflow api for python, version r1.6. Available at `https://www.tensorflow.org/api_docs/python/`.

Breiman, L. (2001). Random forests. *Machine Learning*, pages 5–32.

Casilaria, E., Santoyo-Ramón, J. A., and Cano-García, J. M. (2017a). Umafall: A multisensor dataset for the research on automaticfall detection. *The 14th International Conference on Mobile Systems and Pervasive Computing*, pages 32–29.

Casilaria, E., Santoyo-Ramóna, J. A., and Cano-Garcíaa, J. M. (2017b). Umafall: A multisensor dataset for the research on automatic fall detection. *The 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017)*, pages 32–39.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, pages 21–27.

Gasparrini, S., Cippitelli, E., Spinsante, S., and Gambi, E. (2014). A depth-based fall detection system using a kinect@ sensor. *Sensors*, page 2756–2775.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, pages 3–42.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gope, P. and Hwang, T. (2016). Bsn-care: A secure iot-based modern healthcare system using body sensor network. *IEEE Sensors Journal*, pages 1368–1376.

JetBrains (2018). Pycharm v2018.1. Available at `https://www.jetbrains.com/pycharm/download/`.

Karim, F., Majumdar, S., Darabi, H., and Chen, S. (2018). Lstm fully convolutional networks for time series classification. *IEEE Access*, 6:1662–1669.

Katz, S. (1984). Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, 31:721–727.

Khan, A. M., Lee, Y. K., Lee, S. Y., and T.S. Kim, T. K. (2010). Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis. *5th International Conference on Future Information Technology. IEEE*, pages 1–6.

Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. *European Conference on Machine Learning*, pages 4–15.

Mano, L. Y., Faiçal, B. S., Nakamura, L. H. V., Gomes, P. H., Libralon, G. L., Meneguete, R. I., Filho, G. P. R., Giancristofaro, G. T., Pessin, G., Krishnamachari, B., and Ueyama, J. (2016). Exploiting iot technologies for enhancing health smart homes through patient identification and emotion recognition. *Elsevier Computer Communications*, pages 178–190.

Micucci, D., Mobilio, M., and Napoletano, P. (1995). Support-vector networks. *Springer Netherlands*, pages 273–297.

Micucci, D., Mobilio, M., and Napoletano, P. (2016). Unimib shar: a new dataset for human activity recognition using acceleration data from smartphones. *IEEE Sensors Lett*, pages 15–18.

Mulak, P. and Talhar, N. (2015). Analysis of distance measures using k-nearest neighbor algorithm on kdd dataset. *International Journal of Science and Research (IJSR)*, pages 2101–2104.

Ojetola, O., Gaura, E., and Brusey, J. (2015). Data set for fall events and daily activities from inertial sensors. *Proceedings of the 6th ACM Multimedia Systems Conference (MMSys'15)*, pages 243–248.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, pages 2825–2830.

Python (2017). Python language reference, version 3.6.4. Available at `https://docs.python.org/release/3.6.4/`.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, pages 81–106.

Shahmohammadi, F., Hosseini, A., King, C. E., and Sarrafzadeh, M. (2017). Smartwatch based activity recognition using active learning. *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 321–329.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning:From Theory to Algorithms*. Cambridge University Press.

Smola, A. and Vishwanathan, S. (2008). *Introduction to Machine Learning*. Cambridge University Press.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, pages 111–147.

Sucerquia, A., López, J., and Vargas-bonilla, J. (2017). Sisfall: A fall and movement dataset. *Sensors (Basel)*, pages 1–14.

Vavoulas, G., Chatzaki, C., Malliotakis, T., and Pediaditis, M. (2016). The mobiact dataset: Recognition of activities of daily living using smartphones. *Proceedings of the International Conference on Information and Communication Technologies for Ageing Well and e-Health (ICT4AWE)*, pages 143–151.

Vavoulas, G., Pediaditis, M., Spanakis, E., and Tsiknakis, M. (2013). The mobifall dataset: An initial evaluation of fall detection algorithms using smartphones. *Proceedings of the IEEE 13th International Conference on Bioinformatics and Bioengineering (BIBE 2013)*, pages 1–4.

Weiss, G. M., Timko, J. L., Gallagher, C. M., Yoneda, K., and Schreiber, A. J. (2016). Smartwatch-based activity recognition : A machine learning approach. *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics*, pages 426–429.