

# Avaliação de Técnicas de Classificação\*

Tiago C A Amorim (RA: 100675)<sup>a</sup>, Taylon L C Martins (RA: 177379)<sup>b</sup>

<sup>a</sup>Doutorando no Departamento de Engenharia de Petróleo da Faculdade de Engenharia Mecânica, UNICAMP, Campinas, SP, Brasil

<sup>b</sup>Aluno especial, UNICAMP, Campinas, SP, Brasil

**Keywords:** Classificação, Regressão Logística, k-Vizinhos mais Próximos, Validação Cruzada

## 1. Introdução

Este relatório apresenta as principais atividades realizadas no desenvolvimento das atividades propostas na Lista 02 da disciplina IA048: Aprendizado de Máquina, primeiro semestre de 2024. O foco deste exercício é de construir a avaliar o desempenho de algoritmos de classificação usando duas versões de um mesmo estudo.

## 2. Tarefa Proposta

Nesta atividade, vamos abordar o problema de reconhecimento de atividades humanas (HAR, do inglês *human activity recognition*) a partir de informações capturadas por sensores de smartphones. Em particular, vamos trabalhar com a base de dados UCI HAR [1], que contém registros de sensores inerciais presentes em um smartphone preso à cintura de 30 sujeitos realizando atividades cotidianas. Cada pessoa realizou seis atividades, as quais correspondem aos seguintes rótulos:

Atividade <sup>1</sup>		Rótulo
Caminhar	( <i>Walking</i> )	1
Subir escadas	( <i>W. upstairs</i> )	2
Descer escadas	( <i>W. downstairs</i> )	3
Sentado	( <i>Sitting</i> )	4
Em pé	( <i>Standing</i> )	5
Deitado	( <i>Laying</i> )	6

Tabela 1: Rótulos da base de dados.

Foram capturadas as amostras dos três eixos (x, y e z) do acelerômetro (ACC, do inglês *accelerometer*) e do giroscópio (GYR, do inglês *gyroscope*) presentes no smartphone, empregando uma taxa de amostragem de 50 Hz. O conjunto completo de amostras foi particionado aleatoriamente em treinamento (70% dos voluntários) e teste (30% dos voluntários).

\*Relatório número 02 como parte dos requisitos da disciplina IA048: Aprendizado de Máquina.

<sup>1</sup>Foram adicionados os termos originais entre parênteses para facilitar a comparação com os gráficos, que foram construídos com os termos em inglês.

### 2.1. Primeira parte

Primeiramente, será explorada uma versão do conjunto de dados na qual já houve pré-processamento e extração de características. No caso, cada amostra contém 561 atributos derivados de uma mesma janela de 2,56 s dos 6 sinais disponíveis (ACC: x,y,z; GYR: x,y,z), considerando suas representações tanto no domínio do tempo quanto no domínio da frequência.

- Construa uma solução para este problema baseada no modelo de regressão logística. Descreva a abordagem escolhida para resolvê-lo (softmax, classificadores binários combinados em um esquema um-contrum ou um-contratodos). Obtenha, então, a matriz de confusão para o classificador considerando os dados do conjunto de teste. Além disso, adote uma métrica global para a avaliação do desempenho (médio) deste classificador. Discuta os resultados obtidos.
- Considere, agora, a técnica k-nearest neighbors (kNN). Adotando um esquema de validação cruzada, mostre como o desempenho do classificador, computado com a mesma métrica adotada no item (a) varia em função do parâmetro k. Escolhendo, então, o melhor valor para k, apresente a matriz de confusão para os dados de teste e o desempenho medido nesse conjunto. Comente os resultados obtidos, inclusive estabelecendo uma comparação com o desempenho da regressão logística.

### 2.2. Segunda parte

Agora, vamos utilizar os dados “brutos” combinados de ACC e GYR como entradas dos classificadores. Para isso, devemos recorrer aos registros disponibilizados no diretório ‘Inertial Signals’, os quais estão separados por eixo e por sensor, sendo que cada amostra individual agora é formada por 128 valores (atributos), que correspondem às amplitudes instantâneas de aceleração (ACC) ou velocidade angular (GYR) dentro de uma janela de 2,56 s.

- Monte, então, a nova matriz de entrada concatenando os seis sinais temporais e, então, repita o procedimento experimental detalhado nos itens (a) e (b). Ao final, com base no desempenho obtido, teça uma análise comparativa entre a abordagem do item anterior e

a abordagem baseada nos sinais “brutos” empregada nesta segunda parte.

### 3. Aplicação

Toda a avaliação foi feita em um único *notebook* Jupyter, em Python. Foi feito o uso da biblioteca *Scikit-learn* [2] para fazer as diferentes manipulações nos dados. O código pode ser encontrado em [https://github.com/TiagoCAAmorim/machine\\_learning](https://github.com/TiagoCAAmorim/machine_learning).

#### 3.1. Conjuntos de Dados

Os dados foram disponibilizados em formato tabular, já com uma separação entre os dados de treino e de teste (tabela 2). Os dados pré-processados são formados por 561 atributos, enquanto que os dados brutos são formados por 768 atributos<sup>2</sup>. Uma descrição de cada um dos atributos é feita pelos autores no pacote do conjunto de dados [1].

Conjunto	Número de Amostras
Treino	7 352 (71.4%)
Teste	2 947 (28.6%)
Total	10 299 (100%)

Tabela 2: Tamanho da base de dados.

Aparentemente o desbalanço entre as classes não é significativo, mas existe (figura 1). A menor classe tem cerca de 30% menos amostras que a maior classe. De toda forma será utilizada a **acurácia balanceada** como métrica da qualidade do classificador (média dos *recalls* de cada classe).

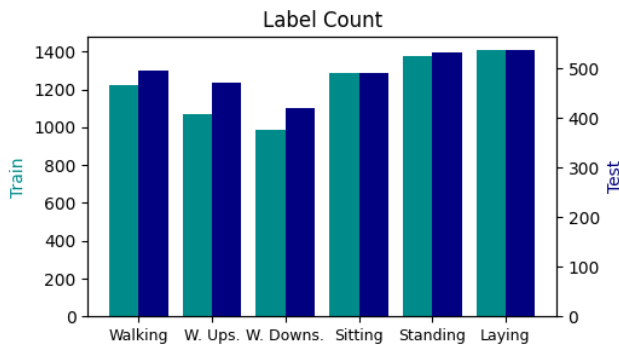


Figura 1: Número de amostras por classe.

#### 3.2. Dados Pré-processados

Os dados pré-processados estão normalizados no intervalo  $[-1; 1]$ , à exceção de alguns dos atributos (figura 2). Desta forma, em um primeiro momento não existe necessidade de normalizar os dados.

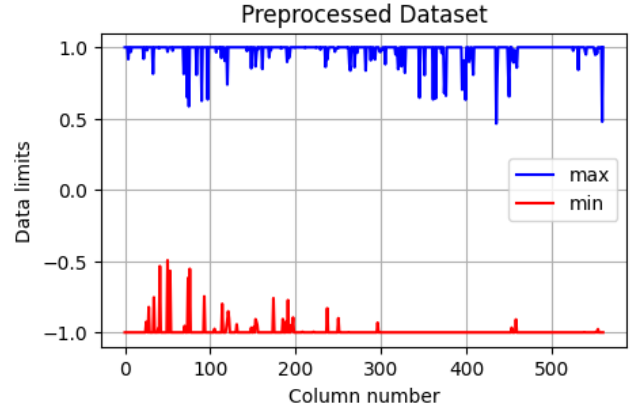


Figura 2: Limites dos atributos do conjunto de treinamento dos dados pré-processados.

#### 3.2.1. Regressão Logística

O classificador de regressão logística foi construído com a classe **LogisticRegressionCV** do *Scikit-Learn*. Esta classe realiza a otimização do parâmetro de regularização junto com validação cruzada. Foram utilizadas as seguintes opções para o ajuste deste modelo:

1. Validação cruzada estratificada em 5 pastas.
2. Normalização do tipo  $l_2$  ( $\frac{1}{2}||w||_2^2$ ), com otimização do seu inverso ( $c = \frac{1}{l_2}$ )
3. Função objetivo da otimização: acurácia balanceada.
4. Estratégia: multinomial (entropia cruzada<sup>3</sup>).

O modelo ajustado tem  $l_2 = 0.3594$ . A acurácia balanceada na validação cruzada foi de 0.9932, e com os dados de teste 0.9598. Observa-se que o classificador tem um bom desempenho, e que o F1-score também seria uma boa escolha para avaliar a qualidade dos classificadores. (tabela 3).

Ao analisar por classe, fica claro que o desempenho não é uniforme. A classe **Sentado** tem um valor de *recall* bem mais baixo que as demais, pois o classificador tem dificuldade em distinguir **Sentado** de **Em pé** (figura 3).

Classe	<i>Recall</i>	<i>F1 score</i>
Todas <sup>4</sup>	0.9598	0.9606
Caminhar	0.9940	0.9686
Subir escadas	0.9427	0.9569
Descer escadas	0.9690	0.9795
Sentado	0.8717	0.9214
Em pé	0.9812	0.9372
Deitado	1.0000	1.0000

Tabela 3: Resultados do classificador com regressão logística para os dados pré-processados.

<sup>2</sup>Neste estudo foram ignorados os dados de aceleração total, que são 384 atributos adicionais.

<sup>3</sup>Equivalente a utilizar *Softmax*.

<sup>4</sup>*Recall* médio é a acurácia balanceada.

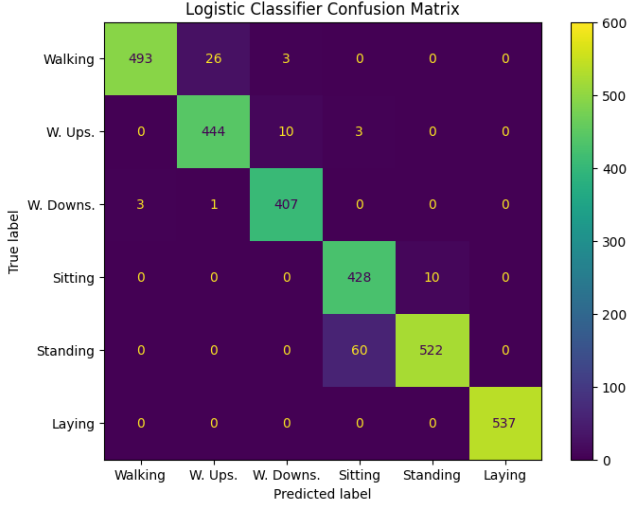


Figura 3: Matriz de confusão do classificador com regressão logística para os dados pré-processados.

### 3.2.2. k-Vizinhos mais Próximos

Na primeira tentativa de construção de um classificador com k-Vizinhos mais Próximos ( $kNN$ ) foram utilizadas as opções padrão da classe `KNeighborsClassifier` do *Scikit-Learn*: distância euclidiana e pesos uniformes. Novamente foi utilizada a validação cruzada estratificada em 5 pastas.

O valor ótimo de  $k$  ficou em 17 (figura 4). Este classificador com opções padrão (*Vanilla*) ficou com acurácia balanceada na validação cruzada igual a 0.8999, e com os dados de teste igual a 0.8999<sup>5</sup>. Este classificador teve resultados inferiores ao do classificador com regressão logística (tabela 4). Os resultados foram piores em todas as classes.

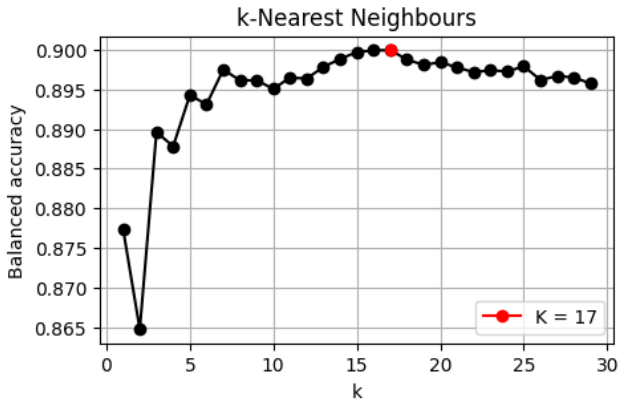


Figura 4: Otimização do parâmetro  $k$  do classificador com k-Vizinhos mais Próximos para os dados pré-processados.

Dada a menor performance do classificador com k-Vizinhos mais Próximos, foram realizados alguns testes para tentar

<sup>5</sup> A coincidência de valores levantou suspeitas quanto ao código desenvolvido. O código foi verificado mais de uma vez, e nenhum erro foi encontrado. Os valores diferem na quinta casa decimal.

Classe	Recall	F1 score
Todas	0.8999	0.9017
Caminhar	0.9839	0.9104
Subir escadas	0.9108	0.9022
Descer escadas	0.7667	0.8530
Sentado	0.7984	0.8578
Em pé	0.9436	0.8885
Deitado	0.9963	0.9981

Tabela 4: Resultados do classificador com k-Vizinhos mais Próximos para os dados pré-processados.

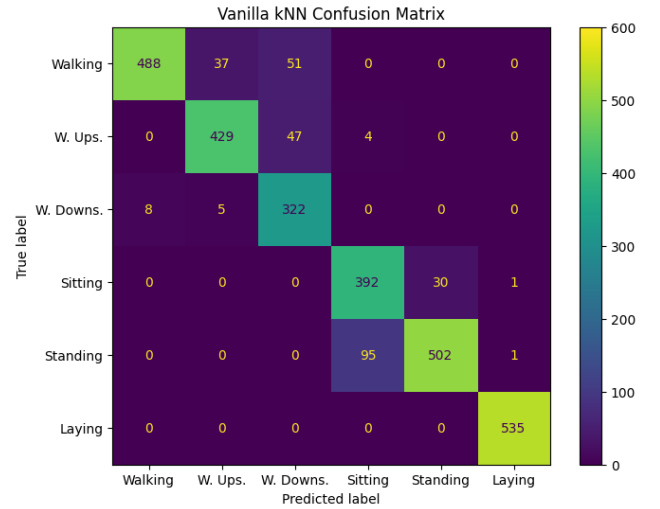


Figura 5: Matriz de confusão do classificador com k-Vizinhos mais Próximos para os dados pré-processados.

melhorar o seu resultado. A primeira tentativa foi uma busca em grade ao redor do melhor  $k$  encontrado, buscando os melhores hiperparâmetros para o classificador de k-Vizinhos mais Próximos. Foram testados diferentes valores de  $p$  para a métrica de Minkowski e o uso de pesos ponderados pelo inverso da distância (tabela 5). O classificador com os hiperparâmetros ótimos ( $kNN_{opt1}$ ) mostrou um pequeno ganho na acurácia balanceada: 0.9045 na validação cruzada e 0.9146 nos dados de teste.

Hiperparâmetro	Padrão ( <i>Vanilla</i> )	Otimizado
Distância	Euclidiana	Manhattan
Pesos	Uniforme	Distância

Tabela 5: Hiperparâmetros do classificador k-Vizinhos mais Próximos.

Como alguns dos dados de entrada não estava *exatamente* escalados em  $[-1;1]$ , a segunda tentativa foi aplicar uma escala  $[0;1]$  aos dados pré-processados. O resultado foi um pequeno incremento na acurácia balanceada: 0.9059 na validação cruzada e 0.9153 nos dados de teste.

A última etapa na busca por um classificador com k-

Vizinhos mais Próximos de melhores resultados foi avaliar os parâmetros de entrada. Foi feita uma otimização *gulos* dos parâmetros que são utilizados na construção do classificador de k-Vizinhos mais Próximos. A cada iteração são avaliados todos os parâmetros de entrada. A cada passo é retirado um dos parâmetros de entrada e o modelo é reconstruído. Se a acurácia balanceada aumentar, este parâmetro é retirado permanentemente. O algoritmo termina quando nenhum parâmetro é retirado em uma iteração.

Para este estudo foi feita apenas uma iteração, ou seja, a retirada de cada atributo foi testada uma única vez. Este procedimento excluiu 65 dos 561 atributos, e levou a uma acurácia balanceada de 0.9218 na validação cruzada e de 0.9192 nos dados de teste.

Uma possível etapa adicional seria a otimização do *peso* de cada atributo. Este procedimento seria a otimização de 496 pesos. Devido ao custo computacional envolvido em uma otimização de tantos parâmetros, optou-se por considerar a avaliação concluída.

Foi possível melhorar os resultados com relação a otimizar apenas o número de vizinhos, mas o incremento foi relativamente pequeno. Os resultados com os dados de teste geralmente seguiram os incrementos nos resultados com a validação cruzada. Em todas as tentativas os resultados ficaram abaixo daqueles do classificador com regressão logística.

### 3.3. Dados Brutos

Os dados brutos de aceleração e do giroscópio nas três direções levam a um conjunto de dados com 768 atributos. Os dados não estão normalizados (figura 6). Os dados pré-processados estavam *aproximadamente* em  $[-1; 1]$  e foi pequeno, para o modelo de k-Vizinhos mais Próximos, o ganho de mudar a escala para  $[0; 1]$ . Os dados brutos foram escalados para  $[-1; 1]$ .

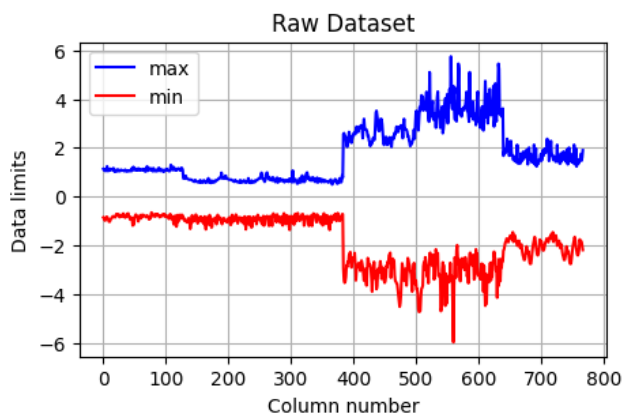


Figura 6: Limites dos atributos do conjunto de treinamento dos dados brutos.

#### 3.3.1. Regressão Logística

#### 3.3.2. k-Vizinhos mais Próximos

### 4. Conclusão

### Referências

- [1] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, et al., A public domain dataset for human activity recognition using smartphones., in: Esann, Vol. 3, 2013, p. 3.
- [2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.