Human Activity Recognition using Accelerometer Data from Wearable Sensors

Miguel Herrera Cooper
UNSA - Ciencia de la Computación
Arequipa, Perú
mherreraco@unsa.edu.pe

Yordy Santos Apaza
UNSA - Ciencia de la Computación
Arequipa, Perú
ysantosa@unsa.edu.pe

Milagros Cruz Mamani
UNSA - Ciencia de la Computación
Arequipa, Perú
mcruzmam@unsa.edu.pe

Alexander Cordova Ccana
UNSA - Ciencia de la Computación
Arequipa, Perú
acordovacc@unsa.edu.pe

ABSTRACT

En este trabajo, implementamos y evaluamos un algoritmo de clasificación para detectar cuatro actividades físicas humanas cruciales (caminar, andar en bicicleta, sentarse y acostarse) utilizando un dataset de cinco acelerómetros triaxiales usados simultáneamente en diferentes partes del cuerpo (cadera dominante, parte superior del brazo, tobillo, muslo y muñeca). Los datos del acelerómetro se recopilaron, limpiaron y preprocesaron para extraer características de una ventana de 10 s. Estas características del dominio del tiempo y la frecuencia se usaron con Random Forest y k-Nearest Neighbour para clasificar las actividades. Para la validación los algoritmos se evaluaron con Leave-One-Subject-Out (LOSO).

1 INTRODUCTION

La detección automática de actividades físicas humanas podría tener un gran impacto no solo en las intervenciones de salud, las redes sociales, el estilo de vida, sino también en la publicidad dirigida y la gestión empresarial. Los acelerómetros se pueden utilizar como detectores de movimiento, así como para la detección de la posición corporal y la postura. Su bajo consumo de energía, sus pequeñas dimensiones y su peso ligero los convierten en excelentes candidatos para hacer que el monitoreo de actividad a largo plazo, la evaluación del equilibrio y la detección de caídas sean más prácticos.

Estos sensores se pueden usar en uno o varios sitios del cuerpo. Es preferible una configuración multisitio para detectar una variedad de actividades con una complejidad fina al capturar el movimiento de la parte superior e inferior del cuerpo de forma independiente. La ubicación de la cadera se ha estudiado ampliamente en el reconocimiento de actividad, ya que generalmente captura los principales movimientos corporales.

En este trabajo, implementamos algoritmos de clasificación para datos de acelerómetro de laboratorio previamente recopilados de 33 participantes que realizaban un conjunto de actividades. Nuestro objetivo es clasificar las actividades para cuatro clases (caminar, andar en bicicleta, sentarse y acostarse), utilizando una LOSO para evaluar el rendimiento del algoritmo. A partir de una ventana de 10 segundos, extrajimos algunas de las características más comunes en la clasificación de actividades según los datos del acelerómetro.

2 DATASET

El Dataset consta de 33 participantes reclutados en la comunidad de Stanford, California, para otros estudios de investigación de los doctores William Haskell y Mary Rosenberg. [2]

- https://www.dropbox.com/sh/i3fgtw3zzr4r0ih/AAApsAIat0tF_xo1locMD8ara

2.1 Descripción de los datos

El conjunto de datos consta de varios archivos por participante, entre estos archivos, sólo nos interesan seis archivos:

- 5 archivos son los diferentes archivos de salida de cada sensor colocado en diferentes lugares.
- El sexto archivo es el archivo de intervalo de anotación (AnnotationIntervals.csv).

Los archivos antes mencionados son:

- AnnotationIntervals.csv
- Wocket_00_DatosDominantesCorregidos.csv
- Wocket_01_DatosRefutados_MusloDominante.csv
- Wocket_02_DatosRayosCorregidos_CaderaDominante.csv
- Wocket_03_DatosRayosCorregidos_MuñecaDominante.csv
- Wocket_04_DatosDominantesCorregidos.csv

Los archivos AnnotationIntervals.csv tienen los intervalos de tiempo y las anotaciones de actividad y los archivos wocket tienen la aceleración x,y,z junto con la marca de tiempo.

2.2 Preprocesamiento de los datos

Para el preprocesamiento, se extrajo los archivos mencionados de cada participante y se creo un archivo combinado. Este archivo combinado se crea para cada participante. Para la fusión, tomamos cada archivo wocket con la marca de tiempo y lo fusionamos con el archivo de anotación colocando cada marca de tiempo en el intervalo de tiempo correcto. En el archivo de anotaciones añadimos 2 segundos a la hora de inicio y restamos 2 segundos a la hora de finalización para evitar la fase de transición entre dos actividades. La fase de transición se descartó para que los datos resultantes fueran lo más inequívocos posible en cuanto a su etiqueta de actividad. El archivo final fusionado tiene las columnas X,Y,Z, Ubicación del sensor, Sello de tiempo y Actividad. Esto se realizó para los datos de los 33 participantes.

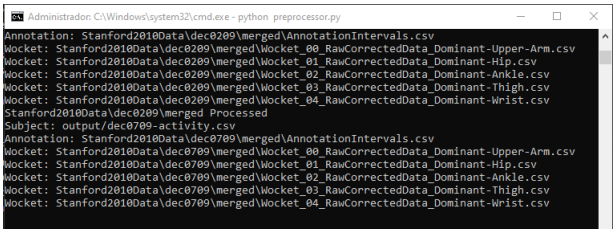

Figure 1: Preprocesamiento de los datos

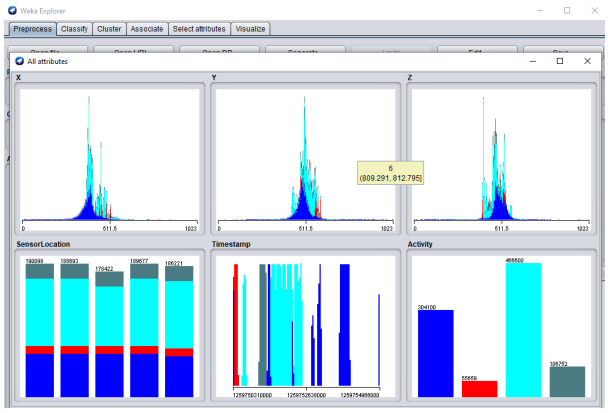

Figure 2: Observación de los datos

3 PATRONES KNN Y RANDOM FOREST (USANDO GRIDSEARCH)

Durante la ejecución del proyecto se usaron 2 algoritmos en específico los cuales son KNN (K-Nest-Neighbors) y RF (Random Forest), estos últimos mostraron tener los mejores resultados para la clasificación de datos, luego de haber sido testeados junto a otros algoritmos de clasificación como: SVM, One vs Rest, Regresión logística, LDA, entre otros.

3.1 Algoritmo KNN

En el proyecto KNN ejecuto un papel muy importante a la hora de la clasificación y selección de los datos, ya que al filtrar los datos irrelevantes se pudo obtener una mejor clasificación y predicción de los mismos. KNN es un aprendizaje basado en instancias o aprendizaje perezoso, en el cual la función sólo se aproxima localmente y todo el cálculo se aplazó hasta la parte de la clasificación, debido a que KNN es uno de los algoritmos más sencillos y con menor complejidad O(n), n es igual al número de puntos de datos, fue fácilmente usado e implementado en la realización del proyecto.

KNN es un algoritmo de aprendizaje supervisado en el que el resultado de la consulta de una nueva instancia se clasifica basándose en la mayoría de categorías KNN y al ser uno de los algoritmos de clasificación y reconocimiento de patrones más conocidos fue fácil de implementar en el proyecto, pero aunque KNN [1] sea sencillo de entender e implementar, depende mucho de obtener el mejor valor para K, por ejemplo, los valores más grandes de K reducen el efecto de ruido durante la clasificación, pero hacen que los límites entre

```
import numpy as np
import pandas as pd
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.grid_search import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
```

Figure 3: Librerías usadas en Python3 para clasificación con KNN

las clases sean menos nítidos. Se puede obtener un valor óptimo de K mediante varias técnicas heurísticas, por ejemplo, la validación cruzada, esta última fue usada durante la ejecución del proyecto como se aprecia en la siguiente imagen.

```
print "Beginning 10-Fold cross validation on all the 33 subjects for sensor at wrist position..."
clf_cross_val = KNeighborsClassifier(n_neighbors=9, algorithm='auto', weights='uniform')
scores = cross_val_score(clf_cross_val, train_data_wrist,target_label_wrist,cv=10)
print "Scores for each fold:"
print scores
print "--------------------------------------------------------------------------------"
print ("Accuracy for 10Fold cross validation using KNN: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
print "\n"
```

Figure 4: Uso cross validation

La fase de entrenamiento de KNN [1] consistió únicamente en almacenar los vectores de características obtenidas y las etiquetas de clase de las muestras de entrenamiento. Durante la fase de clasificación, se usó Gridsearch para la obtener un rango de valores óptimos para K y así obtener los mejores resultados posibles. KNN fue usado en la clasificación de las actividades [4] anteriormente mencionadas además de la lectura y clasificación de los datos captados por los sensores de la cintura, del brazo y otros sensores ubicados en otras áreas del cuerpo de los participantes, también participó en la lectura de los datos de entrenamiento.

```
knn_wrist = KNeighborsClassifier(n_neighbors=3, algorithm='auto', weights='uniform')
param_grid = {
    'n_neighbors': range(3,11,2),
    'algorithm': ['auto', 'ball_tree','kd_tree'],
    'weights': ['uniform','distance'],
}
knn_wrist_gs = GridSearchCV(knn_wrist, param_grid=param_grid)
knn_wrist_gs.fit(train_data_wrist, target_label_wrist)
predicted = knn_wrist_gs.predict(test_data_wrist)
print "Best parameters to be used for training the model",knn_wrist_gs.best_params_
print "\n"
```

Figure 5: Uso de KNN y Gridsearch en Python3

3.2 Algoritmo RF

El algoritmo Random Forest o también conocido como bosques aleatorios está basado en los árboles de decisión y es un algoritmo de aprendizaje por conjuntos para la clasificación, que funcionan construyendo múltiples conjuntos de árboles de decisión y esta construcción es pseudo aleatoria, esa es la razón por la cual se le llama bosque aleatorio, este proceso de creación se da en el momento del entrenamiento y emitiendo la clase, esa es la modalidad en que se desarrollan las clases. RF es veloz y se ejecuta de manera

eficiente en grandes bases de datos, además los bosques generados pueden reusarse en otros datos sin tener que ajustarse en exceso y su complejidad es O(M(mn log n)), siendo n las instancias, m los atributos y M el número de árboles creados.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.grid_search import GridSearchCV
```

Figure 6: Librerías usadas en Python3 para clasificación con RF

Se usó RF debido a que corrigen el sobreajuste habitual en los árboles de decisión individuales, esto se puede lograr gracias a la técnica boostrap o bagging [3], RF crea múltiples arboles de clasificación, con el fin de clasificar un nuevo objeto a partir de un vector de entrada, este vector es colocado en cada árbol del bosque, cada árbol da una clasificación sobre el vector y se puede decir que el árbol votó por esa clase, de esa forma el bosque elige la clasificación que tiene más votos en todo el bosque. Para brindarle mayor robustez al algoritmo se debe seleccionar los valores más óptimos para los valores de RF por lo cual se hizo uso de Gridsearch al igual que en KNN para obtener una clasificación mucho más eficiente como se ve en la imagen.

```
# clf = RandomForestClassifier(n_estimators=100, criterion='entropy')
# clf.fit(train_data_wrist, target_label_wrist)
# predicted = clf.predict(test_data_wrist)
rfc_wrist = RandomForestClassifier(n_estimators=100, criterion='entropy',n_jobs=-1)
param_grid = {
    'n_estimators': [50, 100, 200],
    'criterion': ['entropy', 'gini']
}
rfc_wrist_gs = GridSearchCV(rfc_wrist, param_grid=param_grid)
rfc_wrist_gs.fit(train_data_wrist, target_label_wrist)
predicted = rfc_wrist_gs.predict(test_data_wrist)
# print "Best parameters to be used for training the model",rfc_wrist_gs.best_params_
```

Figure 7: Uso de RF y Gridsearch en Python3

3.3 GridSearch

El uso de Gridsearch para el desarrollo del proyecto fue indispensable para restringir el rango de valores necesarios para los algoritmos anteriormente descritos, esta restricción de rango nos ayudó a poder limitar los valores necesarios y por lo tanto optimizó los resultados. Gridsearch aceptó un rango de parámetros que fueron usados para ajustar los datos y este a su vez puede ser optimizado por la búsqueda cruzada, usando ambos métodos se llegó a obtener los siguientes rangos de valores para KNN y RF.

- KNN:
  - El rango de K es [9-11] y el peso es uniforme para todos los puntos.
  - Cálculo de las distancias: euclidiana.
- RF:
  - Rango de árboles de decisión generados es [50-200].
  - Algoritmo empleado: InfoGain (es una entropía)[5].

4 RESULTADOS Y CONCLUSIÓN

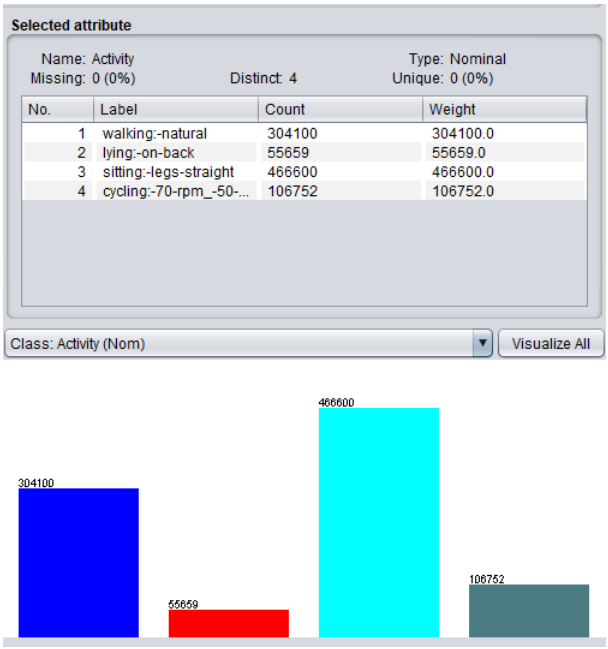

Figure 8: Visualización de los datos

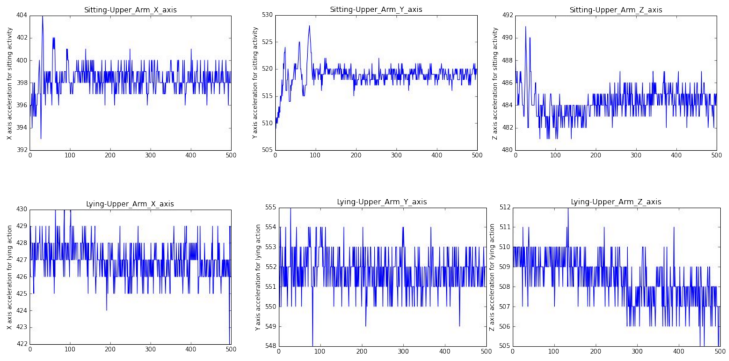

Figure 9: Trazado de los valores de aceleración x, y, z para actividades sentado y acostado

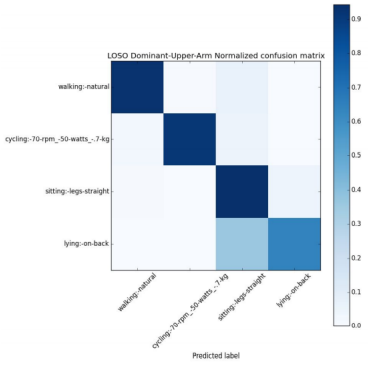

Figure 10: Matriz de confusión despues de LOSO con Random Forest

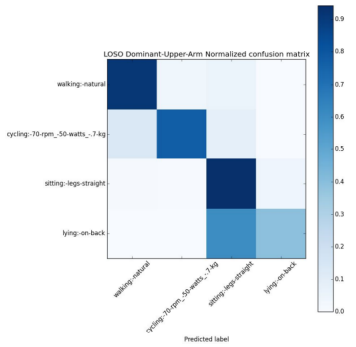

Figure 11: Matriz de confusión despues de LOSO con k-NN

4.1 Conclusiones

Hemos identificado las características más adecuadas requeridas para la clasificación de las actividades mencionadas anteriormente: - Media, desviación de St., mediana, junto con características del dominio de frecuencia extraídas del vector de magnitud de la señal. Experimentos anteriores han demostrado que los datos del acelerómetro triaxial se pueden utilizar para clasificar este tipo de actividades humanas.

Realizamos LOSO y técnicas de validación cruzada de 10 veces para validar nuestro modelo utilizando Random Forest y k-NN, y para ambos, RF funciona mejor que k-NN. También encontramos que el sensor ubicado en la posición dominante de la cadera fue el más discriminativo en la clasificación, seguido del sensor en el tobillo dominante. Otra tarea que se realizó fue combinar datos de dos sensores ubicados en dos posiciones diferentes y realizar la LOSO y el CV de 10 veces.

Esto resultó en una precisión de clasificación mejorada para algunas de las actividades que no se clasificaron como sólidamente utilizando datos de un solo sitio. El trabajo futuro podría incluir la inclusión de actividades humanas más finas en la clasificación e implementar el algoritmo en un sistema en tiempo real como un teléfono inteligente. El tamaño de la ventana también podría reducirse de 10 segundos para reducir la latencia de dicho sistema en tiempo real. La detección automática de sitios de colocación de sensores podría reducir el riesgo de que las personas utilicen sensores portátiles de manera inapropiada e ineficiente.

REFERENCES

[1] Mohamed Elhosenyc Aboul Ella Hassanien Alaa Tharwat, Hani Mahdi. 2018. Recognizing human activity in mobile crowdsensing environment using optimized k-NN algorithm. *Expert systems with applications* 107, 1 (Oct. 2018), 32–44. https://doi.org/10.1016/j.eswa.2018.04.017

[2] Andrea Mannini, Stephen S Intille, Mary Rosenberger, Angelo M Sabatini, and William Haskell. 2013. Activity recognition using a single accelerometer placed at the wrist or ankle. *article-journal* 45, 1 (Nov. 2013), 2193–203. https://doi.org/10.1249/MSS.0b013e31829736d6

[3] Md. Azher Uddin Md. Taufeeq Uddin. 2015. Human Activity Recognition from Wearable Sensors using Extremely Randomized Trees. *Engineering and information communication technology* 1 (May 2015), 21–23. https://doi.org/10.1109/ICEEICT.2015.7307384

[4] Charissa Ann Ronao and Sung-Bae Cho. 2017. Recognizing human activities from smartphone sensors using hierarchical continuous hidden Markov models. *International Journal of distributed sensor networks* 13, 1 (Jan. 2017), 16. https://doi.org/10.1177/1550147716683687

[5] Ensar Arif Sagbas Serkan Balli and Musa Peker. 2018. Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm. *Measurement and control* 52, 1 (Nov. 2018), 1–2. https://doi.org/10.1177/0020294018813692