НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Прикладной искусственный интеллект

Лабораторная работа № 1

Выполнил:

Нестеров Иван Алексеевич

Группа № R3237

Преподаватели:

Михаил Александрович Каканов,

Олег Александрович Евстафьев

г. Санкт-Петербург

2021

Распознавание активности человека на основе данных с мобильных сенсоров

Необходимо по данным с мобильных сенсоров при помощи прикладных алгоритмов машинного обучения предсказать активность человека по шести классам движений:

- Движется по прямой
- Движется вверх (например, движение по лестнице вверх)
- Движется вниз (например, движение по лестнице вниз)
- Сидит
- Стоит
- Лежит

Сведения о наборе данных

Набор данных содержит записи датчиков со сматфонов (акселерометр и гироскоп с частотой дискретизации 50 Гц) от 30 участников, выполняющих следующие действия: ходьба, ходьба по лестнице, ходьба по лестнице, сидение, стояние и лежание. Данные были предварительно обработаны при помощи фильтров шума. Набор данных представлен Хорхе Л. Рейес-Ортисом.

Признаки были извлечены из 3-х осевых необработанных сигналов акселерометра и гироскопа tAcc-XYZ и tGyro-XYZ. Эти сигналы были сняты с постоянной частотой 50 Гц. Затем были отфильтрованны с помощью медианного фильтра и низкочастотного фильтра Баттерворта 3-го порядка с частотой 20 Гц для удаления шумов. Аналогичным образом сигнал ускорения был разделен на сигналы ускорения тела и гравитации (`tBodyAcc-XYZ` и `tGravityAcc-XYZ`) с помощью другого низкочастотного фильтра Баттерворта с угловой частотой 0,3 Гц. Линейное ускорение тела и угловая скорость были использованы для получения сигналов "рывка" — (`tBodyAccJerk-XYZ` и `tBodyGyroJerk-XYZ`). Также величина этих трехмерных сигналов была расчитана с использованием евклидовой нормы — (`tBodyAccMag` , `tGravityAccMag` , `tBodyAccJerkMag` , `tBodyGyroMag` , `tBodyGyroJerkMag`).

Наконец, к некоторым из этих сигналов было применено быстрое преобразование Фурье (БПФ), в результате чего получились `fBodyAcc-XYZ` , `fBodyAccJerk-XYZ` , `fBodyGyro-XYZ` , `fBodyAccJerkMag` , `fBodyGyroMag` , `fBodyGyroJerkMag` . (Обратите внимание на "f" для обозначения сигналов в частотной области).

Набор переменных, которые были оценены по этим сигналам, следующий:

- mean(): Среднее значение
- std(): Стандартное отклонение
- mad(): Среднее абсолютное отклонение
- max(): Наибольшее значение в массиве
- min(): Наименьшее значение в массиве
- sma(): Область величины сигнала
- energy(): Мера энергии. Сумма квадратов, деленная на количество значений.
- iqr(): Интерквартильный размах
- entropy(): Энтропия сигнала
- arCoeff(): Коэффициенты авторегрессии с порядком Burg, равным 4
- correlation(): коэффициент корреляции между двумя сигналами
- maxInds(): индекс частотной составляющей с наибольшей величиной

- meanFreq(): средневзвешенное значение частотных компонент для получения средней частоты
- skewness(): перекос сигнала в частотной области
- kurtosis(): эксцесс сигнала в частотной области
- bandsEnergy(): Энергия частотного интервала в пределах 64 бинов БПФ каждого окна.
- angle(): Угол между векторами.

Импорт библиотек

Первым делом импортируем необходимые библиотеки для работы с данными:

```
import os
import numpy as np
import pandas as pd
```

Считываем набор данных

В прикладных задачах машинного обучения очень важен процесс извлечения признаков (feature extraction), в ходе которого данные интерпретируются в информативные признаки. Также этот процесс может называться проектирование признаков (feature engineering), это весьма трудоемкая и творческая задача. В рамках работы мы опустим эту часть и поспользуемся предобработанными данными.

```
def read_data(path, filename):
    return pd.read_csv(os.path.join(path, filename))

df = read_data('/data/notebook_files', 'train.csv')
df.head()
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJe kurtosis()
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205

5 rows × 563 columns

Теперь, загрузим полный набор данных и сохранить его под следующими четырьмя переменными:

- `train_X` : признаки, используемые для обучения модели
- `train_y` : метки, используемые для обучения модели
- `test_X` : признаки, используемые для проверки модели
- `test_y` : метки, используемые для проверки модели

```
def load_dataset(label_dict):
    train_X = read_data('/data/notebook_files', 'train.csv').values[:,:-2]
    train_y = read_data('/data/notebook_files', 'train.csv')['Activity']
    train_y = train_y.map(label_dict).values
    test_X = read_data('/data/notebook_files', 'test.csv').values[:,:-2]
    test_y = read_data('/data/notebook_files', 'test.csv')
    test_y = test_y['Activity'].map(label_dict).values
    return(train_X, train_y, test_X, test_y)
label_dict = {'WALKING':0, 'WALKING_UPSTAIRS':1, 'WALKING_DOWNSTAIRS':2, 'SITTING':3, 'STANDING':4, '
train_X, train_y, test_X, test_y = load_dataset(label_dict)
```

Выбор модели

Импортируйте выбранную вами модель из библиотеки `sklearn` и инициализируйте её в объект `model`:

```
from sklearn import svm
model = svm.SVC(kernel='linear')
```

Оценка модели

Используйте обученную модель для прогнозирования активности движения, используя признаки из тестового набора (`test_X`). Прогнозы сохраните в списке `yhat` .

```
yhat = model.predict(test_X)
yhat
```

Выведите отчет о классификации, сравнив предсказания (`yhat`) с базовой истиной (`test_y`).

```
from sklearn.metrics import classification_report
target_names = ['Walking', 'Walking Upstairs', 'Walking Downstairs', 'Sitting', 'Standing', 'Laying']

print(classification_report(test_y, yhat, target_names=target_names))
```

```
                    precision    recall  f1-score   support

           Walking       0.96      0.99      0.97       496
  Walking Upstairs       0.98      0.96      0.97       471
Walking Downstairs       0.99      0.98      0.98       420
           Sitting       0.96      0.89      0.92       491
          Standing       0.91      0.97      0.94       532
            Laying       1.00      1.00      1.00       537

          accuracy                           0.96      2947
         macro avg       0.97      0.96      0.96      2947
      weighted avg       0.96      0.96      0.96      2947
```

Комментарии к работе:

1. Accuracy – одна из наиболее общих и наименее полезных метрик, переводящийся на русский язык как «точность». Отражает долю правильных ответов. Метрика может быть бесполезна по причине наличия возможности ее искусственного улучшения без улучшения эффективности модели.
2. Precision и recall – метрики, которые позволяют исправить недостатки accuracy и дать действительно информативную оценку работы модели. Precision отражает долю найденных объектов с определенным свойством (по мнению модели), среди объектов, которые этим свойством действительно обладают, а recall – какое количество объектов с этим свойством из всего множества объектов с этим свойством модель нашла. Данные метрики можно обобщить следующим образом: «Precision – сколько выбранных значений правильны?» и «Recall – как много правильных значений выбрано?».
3. F1-score – среднее гармоническое precision и recall, метрика, которая позволяет как-то «объединить» метрики из пункта 2 и остаться при этом информативной.

Мотивация по выбору модели:

В ходе работы много времени занял подбор такой модели, которая выдаст наиболее точный результат. Были взяты различные алгоритмы машинного обучения (из тех, что представлены в лекции №2). Я попробовал метод ближайших соседей, где последовательно менял параметр n_neighbors от 1 до 6, но точность оставалась либо на уровне 0,94, либо падала еще ниже. Далее я попробовал метод линейной регрессии, однако она вообще не запустилась, и я решил не испытывать удачу и попробовать метод опорных векторов. Сначала попробовал параметру kernel задать значение «poly» и изменять значения параметра degree от 1 до 6, но точность лучше, чем 0.95 (лучший результат за все мои попытки) не стала. Далее я решил убрать параметр degree, а значения поля kernel поменять на «linear». Точность выросла до 0.96, а среднее значение precision – до 0.97, чего раньше не было. Полученный результат я счел удовлетворительным и потому завершил поиски лучшего алгоритма.

Ссылка на работу:
https://datalore.jetbrains.com/notebook/U2zWTdpl3R0QTRO9YNQJux/YnZ4hea0tya7tq4Pq2IGUT/

Вывод: в ходе проделанной работы я ознакомился с некоторыми из алгоритмов машинного обучения и опробовал их применение на практике, повторил основы работы со средствами библиотек pandas (сохранение данных в объект и чтение из него).