스마트폰 데이터에서 인간 활동을 모델링하는 방법

2019년 8월 5 일 **Jason Brownlee** 작성 , **시계열에 대한 심층 학습** 92

인간 활동 인식은 특수 하네스나 스마트 폰을 통해 기록된 가속도계 데이터 시퀀스를 잘 알려진, 잘 정의된 움직임으로 분류하는 문제입니다.

매초 생성되는 관측치의 수가 방대하고, 관측치의 시간적 특성이 있으며, 가속도계 데이터를 알려진 움직임과 연관시킬 수 있는 명확한 방법이 부족하다는 점을 감안하면 이는 어려운 문제입니다.

문제에 대한 고전적 접근 방식은 고정된 크기의 윈도우를 기반으로 시계열 데이터에서 피처를 수작업으로 제작하고, 의사 결정 트리 앙상블과 같은 머신 러닝 모델을 훈련하는 것을 포함합니다. 어려움은 이러한 피처 엔지니어링이 해당 분야에 대한 심층적인 전문 지식을 필요로 한다는 것입니다.

최근, 순환 신경망이나 1차원 합성곱 신경망(CNN)과 같은 딥러닝 방법이 데이터 기능 엔지니어링을 거의 또는 전혀 사용하지 않고도 어려운 활동 인식 작업에서 최첨단 결과를 제공하는 것으로 나타났습니다.

이 튜토리얼에서는 시계열 분류를 위한 ' 스마트폰을 사용한 활동 인식 ' 데이터 세트를 알아보고, 예측 모델링에 대비해 데이터 세트를 로드하고 탐색하는 방법을 알아봅니다 .

이 튜토리얼을 완료하면 다음 내용을 알 수 있습니다.

- 데이터 세트를 메모리에 다운로드하고 로드하는 방법.
- 선형 플롯, 히스토그램, 상자 그림을 사용하여 동작 데이터의 구조를 더 잘 이해하는 방법.
- 프레이밍, 데이터 준비, 모델링, 평가를 포함한 문제를 모델링하는 방법입니다.

제가 새로 쓴 '시계열 예측을 위한 딥러닝'이라는 책으로 **프로젝트를 시작해 보세요** . 이 책에는 *단계별 튜토리얼* 과 모든 예제에 대한 *Python 소스 코드 파일이 포함되어 있습니다.*

시작해 보겠습니다.

스마트폰 데이터로 인간 활동을 모델링하는 방법 사진 작가의 사진 , 일부 권리 보유.

튜토리얼 개요

이 튜토리얼은 10개의 부분으로 나뉩니다. 다음과 같습니다.

1. 인간 활동 인식
2. 스마트폰 데이터 세트를 사용한 활동 인식
3. 데이터세트 다운로드
4. 데이터 로드
5. 활동 클래스의 균형

6. 한 주제에 대한 시계열 데이터 플롯
7. 주제별 플롯 히스토그램
8. 활동별 히스토그램 플롯
9. 플롯 활동 기간 상자 그림
10. 모델링에 대한 접근 방식

1. 인간 활동 인식

인간 활동 인식 (HAR)은 센서를 사용하여 사람의 움직임 흔적을 기반으로 사람의 행동을 예측하는 문제입니다.

움직임은 대개 서기, 앉기, 점프하기, 계단 오르기 등 일반적인 실내 활동입니다.

센서는 종종 스마트폰이나 조끼와 같이 대상에 위치하며, 종종 3차원(x, y, z)으로 가속도계 데이터를 기록합니다.

> *인간 활동 인식(HAR)은 자신과 주변 환경에 대한 일련의 관찰을 통해 사람이 수행한 행동을 식별하는 것을 목표로 합니다. 환경 또는 신체 착용 센서와 같은 다양한 소스에서 검색된 정보를 활용하여 인식을 달성할 수 있습니다.*

— 스마트폰을 이용한 인간 활동 인식을 위한 퍼블릭 도메인 데이터 세트 , 2013년.

아이디어는 대상의 활동을 인식하고 알게 되면 지능형 컴퓨터 시스템이 도움을 제공할 수 있다는 것입니다.

센서 데이터를 일반적인 방식으로 특정 동작과 연관시킬 수 있는 명확한 분석적 방법이 없기 때문에 어려운 문제입니다. 수집된 센서 데이터의 양이 방대하고(예: 초당 수십 또는 수백 개의 관찰) 이 데이터에서 수작업으로 만든 기능과 휴리스틱을 고전적으로 사용하여 예측 모델을 개발하기 때문에 기술적으로 어렵습니다.

최근에는 딥러닝 방법이 고차 기능을 자동으로 학습하는 능력을 갖추고 있어 HAR 문제에서 성공적으로 검증되었습니다.

> *센서 기반 활동 인식은 수많은 저수준 센서 판독값에서 인간 활동에 대한 심오한 고수준 지식을 추구합니다. 기존의 패턴 인식 접근 방식은 지난 몇 년 동안 엄청난 진전을 이루었습니다. 그러나 이러한 방법은 종종 휴리스틱 수작업 기능 추출에 크게 의존하여 일반화 성능을 방해할 수 있습니다. [...] 최근 딥 러닝의 최근 발전으로 자동 고수준 기능 추출을 수행할 수 있으므로 많은 분야에서 유망한 성능을 달성합니다.*

— 센서 기반 활동 인식을 위한 딥 러닝: 설문 조사

2. 스마트폰 데이터 세트를 활용한 활동 인식

표준적인 인간 활동 인식 데이터 세트는 2012년에 공개된 ' *스마트폰을 이용한 활동 인식* ' 데이터 세트입니다.

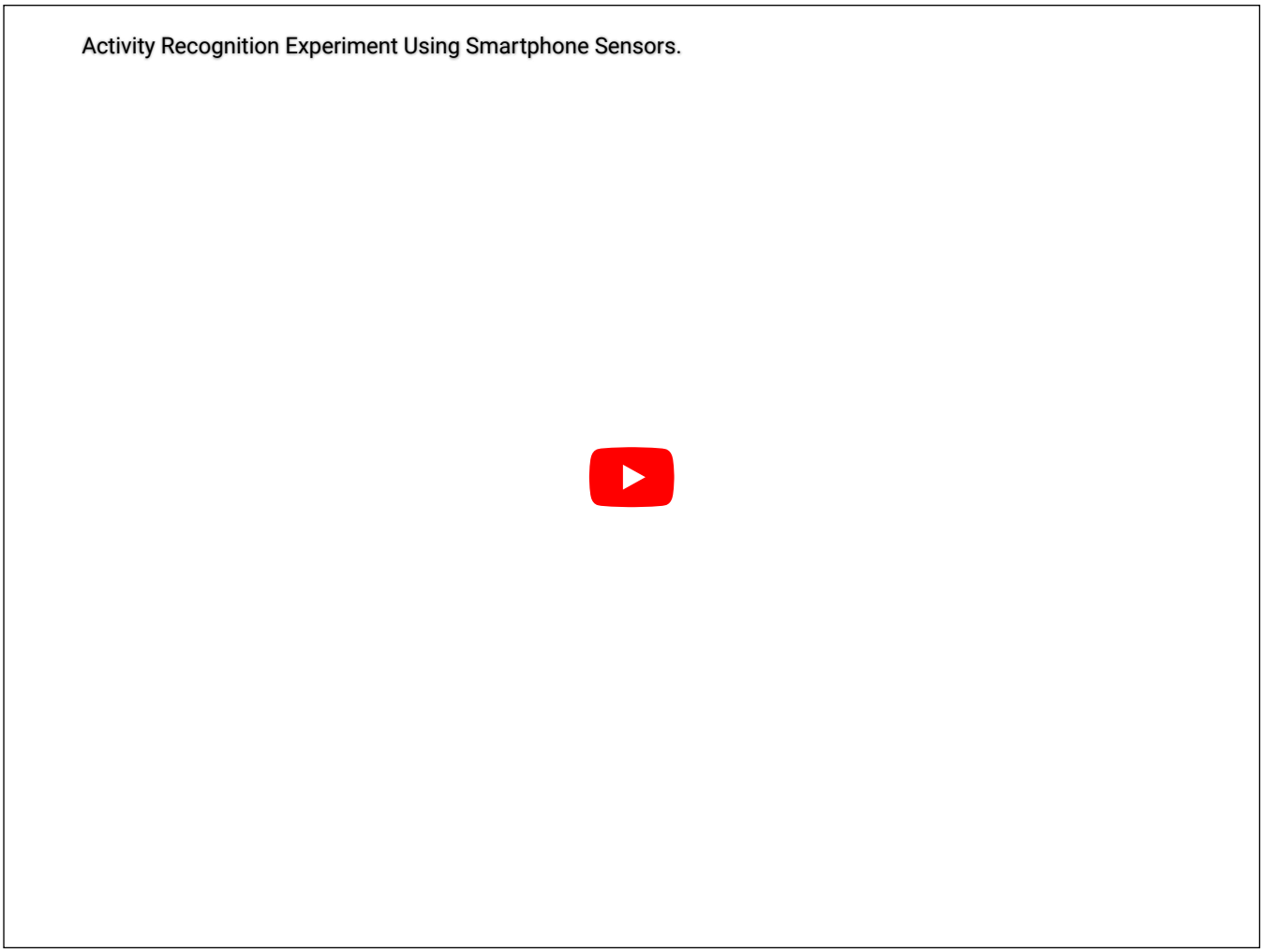
이탈리아 제노바 대학의 Davide Anguita 등이 준비하여 공개했으며, 2013년 논문 " 스마트폰을 사용한 인간 활동 인식을 위한 퍼블릭 도메인 데이터 세트 "에 자세히 설명되어 있습니다. 이 데이터 세트는 2012년 논문 " 다중 클래스 하드웨어 친화적 지원 벡터 머신을 사용한 스마트폰의 인간 활동 인식 " 에서 머신 러닝 알고리즘으로 모델링되었습니다 .

데이터 세트는 UCI Machine Learning Repository에서 무료로 공개되었으며 다운로드할 수 있습니다.

- 스마트폰 데이터 세트를 사용한 인간 활동 인식

데이터는 허리에 장착한 스마트폰을 착용하고 움직임 데이터를 기록하면서 6가지 표준 활동 중 하나를 수행하는 19세에서 48세 사이의 30명의 피험자로부터 수집되었습니다. 각 피험자가 활동을 수행하는 모습을 비디오로 녹화하고 움직임 데이터는 이러한 비디오에서 수동으로 레이블을 지정했습니다.

아래는 피험자가 움직임을 기록하는 동안 활동을 수행하는 모습을 담은 예시 영상입니다.



수행된 6가지 활동은 다음과 같습니다.

1. 걷는
2. 위층으로 걸어 올라가다
3. 아래층으로 걸어가다
4. 좌석
5. 서 있는
6. 부설

기록된 움직임 데이터는 스마트폰, 구체적으로는 Samsung Galaxy S II 의 x, y, z 가속도계 데이터(선형 가속도)와 자이로스코프 데이터(각속도)입니다 .

관찰은 50Hz(즉, 초당 50개 데이터 포인트)로 기록되었습니다. 각 피험자는 활동 순서를 두 번 수행했습니다. 한 번은 장치를 왼쪽에 두고, 한 번은 장치를 오른쪽에 두고 수행했습니다.

> *이 과제를 위해 19세에서 48세 사이의 30명의 자원봉사자 그룹이 선정되었습니다. 각 사람은 허리에 장착한 삼성 갤럭시 S II 스마트폰을 착용한 상태에서 활동 프로토콜을 따르도록 지시받았습니다. 선정된 6가지 ADL은 서기, 앉기, 눕기, 걷기, 계단을 오르기, 오르기였습니다. 각 피험자는 프로토콜을 두 번 수행했습니다. 첫 번째 시도에서는 스마트폰을 벨트의 왼쪽에 고정했고 두 번째 시도에서는 사용자가 선호하는 대로 직접 배치했습니다.*

— 스마트폰을 이용한 인간 활동 인식을 위한 퍼블릭 도메인 데이터 세트 , 2013년.

원시 데이터는 사용할 수 없습니다. 대신, 사전 처리된 버전의 데이터 세트를 사용할 수 있게 되었습니다.

사전 처리 단계는 다음과 같습니다.

- 노이즈 필터를 사용하여 가속도계와 자이로스코프를 사전 처리합니다.
- 50% 중복을 포함하여 2.56초(128개 데이터 포인트)의 고정된 창으로 데이터를 분할합니다.
- 가속도계 데이터를 중력(전체) 구성 요소와 신체 운동 구성 요소로 분리합니다.

> *이러한 신호는 20Hz 차단 주파수를 갖는 3차 저역 통과 버터워스 필터와 중앙 필터로 노이즈 감소를 위해 사전 처리되었습니다. [...] 중력 및 신체 운동 구성 요소가 있는 가속도 신호는 다른 버터워스 저역 통과 필터를 사용하여 신체 가속도와 중력으로 분리되었습니다.*

— 스마트폰을 이용한 인간 활동 인식을 위한 퍼블릭 도메인 데이터 세트 , 2013년.

윈도우 데이터에 피처 엔지니어링이 적용되었으며, 이러한 엔지니어링된 피처가 포함된 데이터 사본이 제공되었습니다.

인간 활동 인식 분야에서 일반적으로 사용되는 여러 시간 및 주파수 특징이 각 창에서 추출되었습니다. 그 결과 특징의 561개 요소 벡터가 생성되었습니다.

데이터 세트는 피험자 데이터를 기준으로 훈련(70%) 세트와 테스트(30%) 세트로 분할되었습니다. 예를 들어, 훈련 피험자는 21명, 테스트 피험자는 9명입니다.

이는 일련의 움직임 활동을 입력으로 사용하여 현재 수행되는 활동의 일부(2.56초)를 예측하고, 알려진 피험자를 대상으로 훈련된 모델을 사용하여 새로운 피험자의 움직임으로부터 활동을 예측하는 문제의 프레이밍을 시사합니다.

스마트폰에서 사용하기 위한 지원 벡터 머신 (예: 고정 소수점 산술)을 사용한 초기 실험 결과 , 테스트 데이터 세트에서 예측 정확도가 89%로 나타났으며, 수정되지 않은 SVM 구현과 유사한 결과를 달성했습니다.

> *이 방법은 표준 지원 벡터 머신(SVM)을 채택하고 고정 소수점 산술을 활용하여 계산 비용을 줄입니다. 기존 SVM과 비교하면 유사한 정확도를 유지하면서도 계산 비용 측면에서 상당한 개선이 나타났습니다. [...]*

— 다중 클래스 하드웨어 친화적 지원 벡터 머신을 사용한 스마트폰에서의 인간 활동 인식 , 2012.

이제 예측 문제에 익숙해졌으므로 이 데이터 세트를 로딩하고 탐색하는 방법을 살펴보겠습니다.

3. 데이터세트 다운로드

이 데이터 세트는 무료로 제공되며 UCI 머신 러닝 저장소에서 다운로드할 수 있습니다.

데이터는 약 58메가바이트 크기의 단일 zip 파일로 제공됩니다. 이 다운로드에 대한 직접 링크는 아래와 같습니다.

- UCI HAR 데이터 세트.zip

데이터 세트를 다운로드하고 모든 파일을 현재 작업 디렉토리의 " HARDataset " 이라는 새 디렉토리에 압축 해제합니다 .

압축 해제된 내용을 검사하면 몇 가지 사항이 눈에 띄게 나타납니다.

- 모델링을 위한 데이터의 분할된 부분(예: 70%/30%)을 포함하는 " *train* " 및 " *test* " 폴더가 있습니다 .
- 데이터 세트에 대한 자세한 기술 설명과 압축 해제된 파일의 내용이 포함된 " *README.txt* " 파일이 있습니다 .
- 엔지니어링된 기능에 대한 기술적 설명이 포함된 " *features.txt* " 파일이 있습니다 .

" train " 및 " *test* " 폴더 의 내용은 비슷합니다(예: 폴더 및 파일 이름). 단, 포함하는 특정 데이터에는 차이가 있습니다.

"train" 폴더를 검사하면 몇 가지 중요한 요소가 표시됩니다.

- 전처리된 데이터가 포함된 " *관성 신호 " 폴더.*
- 모델에 적합하도록 설계된 기능이 포함된 " *X_train.txt " 파일입니다.*
- 각 관찰(1-6)에 대한 클래스 레이블이 포함된 " *y_train.txt "입니다.*
- 데이터 파일의 각 줄과 해당 주체 식별자(1-30)의 매핑이 포함된 " *subject_train.txt " 파일.*

각 파일의 줄 수가 일치합니다. 즉, 각 데이터 파일에서 한 행이 한 레코드임을 나타냅니다.

" *Inertial Signals* " 디렉토리에는 9개의 파일이 있습니다.

- *x, y 및 z 축에 대한 중력 가속도* 데이터 파일: *total_acc_x_train.txt* , *total_acc_y_train.txt* , *total_acc_z_train.txt* .
- *x, y 및 z축에 대한 신체 가속도* 데이터 파일: *body_acc_x_train.txt* , *body_acc_y_train.txt* , body_acc_z_train.txt.
- *x, y 및 z축에 대한 바디 자이로스코프* 데이터 파일: *body_gyro_x_train.txt* , *body_gyro_y_train.txt* , *body_gyro_z_train.txt* .

구조는 " *test* " 디렉토리에 반영됩니다.

우리는 도메인별 기능 엔지니어링을 사용하는 대신 적절한 표현을 학습할 수 있는 머신 러닝 모델을 개발하는 데 가장 흥미로운 " *관성 신호* " 의 데이터에 주목하겠습니다 .

데이터 파일을 검사하면 열이 공백으로 구분되어 있고 값이 -1, 1 범위로 조정된 것으로 나타납니다. 이러한 조정은 데이터 세트와 함께 제공된 *README.txt* 파일 의 메모에서 확인할 수 있습니다 .

이제 어떤 데이터가 있는지 알았으니, 이를 메모리에 로드하는 방법을 알아낼 수 있습니다.

4. 데이터 로드

이 섹션에서는 데이터 세트를 메모리에 로드하는 코드를 개발하겠습니다.

먼저, 하나의 파일을 로드해야 합니다.

read_csv() Pandas 함수를 사용하면 단일 데이터 파일을 로드하고 파일에 헤더가 없으며 공백을 사용하여 열을 구분하도록 지정할 수 있습니다.

```
dataframe = read_csv(filepath, header=None, delim_whitespace=True)
```

이것을 load_file() 이라는 함수로 래핑할 수 있습니다 . 이 함수의 전체 예는 아래에 나와 있습니다.

```
# load dataset
from pandas import read_csv

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

data = load_file('HARDataset/train/Inertial Signals/total_acc_y_train.txt')
print(data.shape)
```

예제를 실행하면 ' total_acc_y_train.txt ' 파일이 로드되고 , NumPy 배열이 반환되고, 배열의 모양이 출력됩니다.

훈련 데이터는 7,352개의 행 또는 데이터 창으로 구성되어 있고, 각 창에는 128개의 관측치가 있는 것을 볼 수 있습니다.

```
(7352, 128)
```

다음으로, 모든 신체 가속도 데이터 파일을 하나의 그룹으로 로드하는 것과 같은 파일 그룹을 로드하는 것이 유용할 것입니다.

이상적으로 다변량 시계열 데이터로 작업할 때 다음 형식으로 데이터를 구조화하는 것이 유용합니다.

```
[samples, timesteps, features]
```

이는 분석에 도움이 되며, 합성곱 신경망이나 순환 신경망과 같은 딥 러닝 모델에서 기대되는 기능입니다.

이는 그룹의 각 파일에 대해 한 번씩, 위의 *load_file()* 함수를 여러 번 호출하여 구현할 수 있습니다 .

각 파일을 NumPy 배열로 로드한 후에는 세 개의 배열을 모두 결합하거나 쌓을 수 있습니다. dstack() NumPy 함수를 사용하여 각 배열이 우리가 선호하는 대로 피처가 3차원에서 분리되도록 쌓이도록 할 수 있습니다.

load_group() 함수는 파일 이름 목록에 대해 이러한 동작을 구현하며 아래에 나열되어 있습니다.

```
# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded
```

전체 가속 파일을 모두 로드하여 이 기능을 시연할 수 있습니다.

전체 예는 아래와 같습니다.

```
# load dataset
from numpy import dstack
from pandas import read_csv

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded

# load the total acc data
filenames = ['total_acc_x_train.txt', 'total_acc_y_train.txt', 'total_acc_z_train.txt']
total_acc = load_group(filenames, prefix='HARDataset/train/Inertial Signals/')
print(total_acc.shape)
```

예제를 실행하면 반환된 NumPy 배열의 모양이 인쇄되며, 데이터 세트에 대한 세 가지 특징(x, y, z)에 대한 예상 샘플 수와 시간 단계가 표시됩니다.

```
(7352, 128, 3)
```

마지막으로, 지금까지 개발된 두 가지 함수를 사용하여 훈련 및 테스트 데이터 세트에 대한 모든 데이터를 로드할 수 있습니다.

train과 test 폴더의 병렬 구조가 주어지면 주어진 폴더에 대한 모든 입력 및 출력 데이터를 로드하는 새 함수를 개발할 수 있습니다. 이 함수는 로드할 모든 9개 데이터 파일의 목록을 빌드하고, 9개 피처가 있는 하나의 NumPy 배열로 로드한 다음, 출력 클래스가 포함된 데이터 파일을 로드할 수 있습니다.

아래의 load_dataset() 함수는 이 동작을 구현합니다. "train" 그룹 또는 "test" 그룹 에 대해 호출할 수 있으며 문자열 인수로 전달됩니다.

```
# load a dataset group, such as train or test
def load_dataset(group, prefix=''):
    filepath = prefix + group + '/Inertial Signals/'
    # load all 9 files as a single array
    filenames = list()
    # total acceleration
    filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt', 'total_acc_z_'+group+'.txt']
    # body acceleration
    filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt', 'body_acc_z_'+group+'.txt']
    # body gyroscope
    filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt', 'body_gyro_z_'+group+'.txt']
    # load input data
    X = load_group(filenames, filepath)
    # load class output
    y = load_file(prefix + group + '/y_'+group+'.txt')
    return X, y
```

전체 예는 아래와 같습니다.

```
# load dataset
from numpy import dstack
from pandas import read_csv

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded

# load a dataset group, such as train or test
def load_dataset(group, prefix=''):
    filepath = prefix + group + '/Inertial Signals/'
    # load all 9 files as a single array
    filenames = list()
    # total acceleration
    filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt', 'total_acc_z_'+group+'.txt']
    # body acceleration
    filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt', 'body_acc_z_'+group+'.txt']
    # body gyroscope
    filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt', 'body_gyro_z_'+group+'.txt']
    # load input data
    X = load_group(filenames, filepath)
    # load class output
    y = load_file(prefix + group + '/y_'+group+'.txt')
    return X, y

# load all train
trainX, trainy = load_dataset('train', 'HARDataset/')
print(trainX.shape, trainy.shape)
# load all test
testX, testy = load_dataset('test', 'HARDataset/')
print(testX.shape, testy.shape)
```

예제를 실행하면 훈련 및 테스트 데이터 세트가 로드됩니다.

테스트 데이터 세트에 2,947개의 윈도우 데이터 행이 있는 것을 볼 수 있습니다. 예상대로, 학습 및 테스트 세트의 윈도우 크기가 일치하고 각 학습 및 테스트 케이스의 출력(y) 크기가 샘플 수와 일치하는 것을 볼 수 있습니다.

```
(7352, 128, 9) (7352, 1)
(2947, 128, 9) (2947, 1)
```

이제 데이터를 로드하는 방법을 알았으니 탐색을 시작해 보겠습니다.

5. 활동 클래스의 균형

데이터를 처음 확인하는 방법은 각 활동의 균형을 조사하는 것입니다.

우리는 30명의 피험자 각각이 6가지 활동을 모두 수행했다고 믿습니다.

이러한 예상을 확인하면 데이터가 실제로 균형을 이루고 있는지 확인하여 모델링을 더 쉽게 만들 수 있으며, 데이터 세트를 올바르게 로드하고 해석하고 있는지 확인할 수 있습니다.

우리는 출력 변수(예: y 변수)의 분석을 요약하는 함수를 개발할 수 있습니다.

아래의 class_breakdown() 함수는 이 동작을 구현하는데, 먼저 제공된 NumPy 배열을 DataFrame으로 래핑하고, 행을 클래스 값으로 그룹화하고, 각 그룹의 크기(행 수)를 계산합니다. 그런 다음 결과를 요약하여 카운트와 백분율을 포함합니다.

```
# summarize the balance of classes in an output variable column
def class_breakdown(data):
    # convert the numpy array into a dataframe
```

```
    df = DataFrame(data)
    # group data by the class value and calculate the number of rows
    counts = df.groupby(0).size()
    # retrieve raw rows
    counts = counts.values
    # summarize
    for i in range(len(counts)):
        percent = counts[i] / len(df) * 100
        print('Class=%d, total=%d, percentage=%.3f' % (i+1, counts[i], percent))
```

훈련 및 테스트 데이터 세트에서 클래스별 분류를 요약하여 유사한 분류가 있는지 확인한 다음, 그 결과를 결합된 데이터 세트의 분류와 비교하는 것이 유용할 수 있습니다.

전체 예는 아래와 같습니다.

```
# summarize class balance
from numpy import array
from numpy import vstack
from pandas import read_csv
from pandas import DataFrame

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

# summarize the balance of classes in an output variable column
def class_breakdown(data):
    # convert the numpy array into a dataframe
    df = DataFrame(data)
    # group data by the class value and calculate the number of rows
    counts = df.groupby(0).size()
    # retrieve raw rows
    counts = counts.values
    # summarize
    for i in range(len(counts)):
        percent = counts[i] / len(df) * 100
        print('Class=%d, total=%d, percentage=%.3f' % (i+1, counts[i], percent))

# load train file
trainy = load_file('HARDataset/train/y_train.txt')
# summarize class breakdown
print('Train Dataset')
class_breakdown(trainy)

# load test file
testy = load_file('HARDataset/test/y_test.txt')
# summarize class breakdown
print('Test Dataset')
class_breakdown(testy)

# summarize combined class breakdown
print('Both')
combined = vstack((trainy, testy))
class_breakdown(combined)
```

먼저 예제를 실행하면 학습 세트에 대한 세부 내역이 요약됩니다. 각 클래스의 분포가 데이터 세트의 13%에서 19% 사이를 맴돌고 있는 것을 볼 수 있습니다.

테스트 세트와 두 데이터 세트의 결과는 함께 매우 유사해 보입니다.

각 훈련 및 테스트 세트, 그리고 주제별로 클래스 분포가 균형을 이룬다고 가정하고 데이터 세트를 사용하는 것이 안전할 것으로 보입니다.

```
Train Dataset
Class=1, total=1226, percentage=16.676
Class=2, total=1073, percentage=14.595
Class=3, total=986, percentage=13.411
Class=4, total=1286, percentage=17.492
Class=5, total=1374, percentage=18.689
Class=6, total=1407, percentage=19.138

Test Dataset
Class=1, total=496, percentage=16.831
Class=2, total=471, percentage=15.982
Class=3, total=420, percentage=14.252
Class=4, total=491, percentage=16.661
Class=5, total=532, percentage=18.052
Class=6, total=537, percentage=18.222

Both
Class=1, total=1722, percentage=16.720
Class=2, total=1544, percentage=14.992
Class=3, total=1406, percentage=13.652
Class=4, total=1777, percentage=17.254
Class=5, total=1906, percentage=18.507
Class=6, total=1944, percentage=18.876
```

6. 한 주제에 대한 시계열 데이터 플롯

우리는 시계열 데이터를 다루고 있으므로 가져오기 확인은 원시 데이터의 선형 플롯을 만드는 것입니다.

원시 데이터는 변수당 시계열 데이터의 창으로 구성되며, 창에는 50%의 중복이 있습니다. 이는 중복을 제거하지 않는 한 선형 플롯으로 관찰에서 일부 반복을 볼 수 있음을 시사합니다.

위에서 개발한 함수를 사용하여 훈련 데이터 세트를 로드하는 것으로 시작할 수 있습니다.

```
# load data
trainX, trainy = load_dataset('train', 'HARDataset/')
```

다음으로, ' *train* ' *디렉토리에 있는* ' *subject_train.txt* ' 를 로드하여 해당 주제에 대한 행 매핑을 제공할 수 있습니다.

load_file() 함수를 사용하여 이 파일을 로드할 수 있습니다 . 로드한 후에는 *unique()* NumPy 함수를 사용하여 훈련 데이터 세트에서 고유한 주제 목록을 검색할 수도 있습니다.

```
sub_map = load_file('HARDataset/train/subject_train.txt')
train_subjects = unique(sub_map)
print(train_subjects)
```

다음으로, 단일 과목(예: 과목 번호 1)에 대한 모든 행을 검색하는 방법이 필요합니다.

이는 주어진 주제에 속하는 모든 행 번호를 찾은 다음 해당 행 번호를 사용하여 훈련 데이터 세트에서 로드된 X와 y 데이터로부터 샘플을 선택하면 됩니다.

아래의 data_for_subject *()* 함수는 이 동작을 구현합니다. 로드된 학습 데이터, 행 번호와 피험자의 로드된 매핑, 그리고 관심 있는 피험자의 피험자 식별 번호를 가져와서 해당 피험자에 대한 *X* 및 *y* 데이터만 반환합니다.

```
# get all data for one subject
def data_for_subject(X, y, sub_map, sub_id):
    # get row indexes for the subject id
    ix = [i for i in range(len(sub_map)) if sub_map[i]==sub_id]
    # return the selected samples
    return X[ix, :, :], y[ix]
```

이제 한 주제에 대한 데이터가 있으므로 이를 그래프로 나타낼 수 있습니다.

데이터는 중복이 있는 창으로 구성되어 있습니다. 이 중복을 제거하고 주어진 변수에 대한 창을 하나의 긴 시퀀스로 압축하여 직선 플롯으로 직접 그릴 수 있는 함수를 작성할 수 있습니다.

아래의 to_series *()* 함수는 주어진 변수(예: 창 배열)에 대해 이러한 동작을 구현합니다.

```
# convert a series of windows to a 1D list
def to_series(windows):
    series = list()
    for window in windows:
        # remove the overlap from the window
        half = int(len(window) / 2) - 1
        for value in window[-half:]:
            series.append(value)
    return series
```

마지막으로, 우리는 데이터를 그릴 만큼 충분합니다. 우리는 피험자의 9개 변수를 차례로 그릴 수 있고, 활동 수준에 대한 최종 플롯을 그릴 수 있습니다.

각 시리즈는 동일한 수의 시간 단계(x축의 길이)를 가지므로, 각 변수에 대한 하위 플롯을 만들고 모든 플롯을 수직으로 정렬하면 각 변수의 움직임을 비교할 수 있어 유용할 수 있습니다.

아래의 plot_subject *() 함수는 단일 피험자의 X* 및 *y* 데이터 에 대해 이 동작을 구현합니다 . 이 함수는 *load_dataset()* 함수 에서 로드된 것과 동일한 변수 순서(3번째 축)를 가정합니다 . 각 플롯에 조잡한 제목도 추가되어 우리가 보고 있는 것이 무엇인지 쉽게 혼동되지 않습니다.

```
# plot the data for one subject
def plot_subject(X, y):
    pyplot.figure()
    # determine the total number of plots
    n, off = X.shape[2] + 1, 0
    # plot total acc
    for i in range(3):
        pyplot.subplot(n, 1, off+1)
        pyplot.plot(to_series(X[:, :, off]))
        pyplot.title('total acc '+str(i), y=0, loc='left')
        off += 1
    # plot body acc
    for i in range(3):
        pyplot.subplot(n, 1, off+1)
        pyplot.plot(to_series(X[:, :, off]))
        pyplot.title('body acc '+str(i), y=0, loc='left')
        off += 1
    # plot body gyro
    for i in range(3):
        pyplot.subplot(n, 1, off+1)
        pyplot.plot(to_series(X[:, :, off]))
        pyplot.title('body gyro '+str(i), y=0, loc='left')
        off += 1
    # plot activities
    pyplot.subplot(n, 1, n)
    pyplot.plot(y)
    pyplot.title('activity', y=0, loc='left')
    pyplot.show()
```

전체 예는 아래와 같습니다.

```
# plot all vars for one subject
from numpy import array
from numpy import dstack
from numpy import unique
```

```
from pandas import read_csv
from matplotlib import pyplot

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded

# load a dataset group, such as train or test
def load_dataset(group, prefix=''):
    filepath = prefix + group + '/Inertial Signals/'
    # load all 9 files as a single array
    filenames = list()
    # total acceleration
    filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt', 'total_acc_z_'+group+'.txt']
    # body acceleration
    filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt', 'body_acc_z_'+group+'.txt']
    # body gyroscope
    filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt', 'body_gyro_z_'+group+'.txt']
    # load input data
    X = load_group(filenames, filepath)
    # load class output
    y = load_file(prefix + group + '/y_'+group+'.txt')
    return X, y

# get all data for one subject
def data_for_subject(X, y, sub_map, sub_id):
    # get row indexes for the subject id
    ix = [i for i in range(len(sub_map)) if sub_map[i]==sub_id]
    # return the selected samples
    return X[ix, :, :], y[ix]

# convert a series of windows to a 1D list
def to_series(windows):
    series = list()
    for window in windows:
        # remove the overlap from the window
        half = int(len(window) / 2) - 1
        for value in window[-half:]:
            series.append(value)
    return series

# plot the data for one subject
def plot_subject(X, y):
    pyplot.figure()
    # determine the total number of plots
    n, off = X.shape[2] + 1, 0
    # plot total acc
    for i in range(3):
        pyplot.subplot(n, 1, off+1)
        pyplot.plot(to_series(X[:, :, off]))
        pyplot.title('total acc '+str(i), y=0, loc='left')
        off += 1
    # plot body acc
    for i in range(3):
        pyplot.subplot(n, 1, off+1)
        pyplot.plot(to_series(X[:, :, off]))
        pyplot.title('body acc '+str(i), y=0, loc='left')
        off += 1
    # plot body gyro
    for i in range(3):
        pyplot.subplot(n, 1, off+1)
        pyplot.plot(to_series(X[:, :, off]))
        pyplot.title('body gyro '+str(i), y=0, loc='left')
        off += 1
    # plot activities
    pyplot.subplot(n, 1, n)
    pyplot.plot(y)
    pyplot.title('activity', y=0, loc='left')
    pyplot.show()

# load data
trainX, trainy = load_dataset('train', 'HARDataset/')
# load mapping of rows to subjects
sub_map = load_file('HARDataset/train/subject_train.txt')
train_subjects = unique(sub_map)
print(train_subjects)
# get the data for one subject
sub_id = train_subjects[0]
subX, suby = data_for_subject(trainX, trainy, sub_map, sub_id)
print(subX.shape, suby.shape)
# plot data for subject
plot_subject(subX, suby)
```

예제를 실행하면 훈련 데이터 세트에 있는 고유한 피험자, 첫 번째 피험자의 데이터 샘플이 인쇄되고, 9개의 입력 변수와 출력 클래스 각각에 대해 하나씩, 총 10개의 플롯이 있는 그림 하나가 생성됩니다.

```
[ 1 3 5 6 7 8 11 14 15 16 17 19 21 22 23 25 26 27 28 29 30]
(341, 128, 9) (341, 1)
```

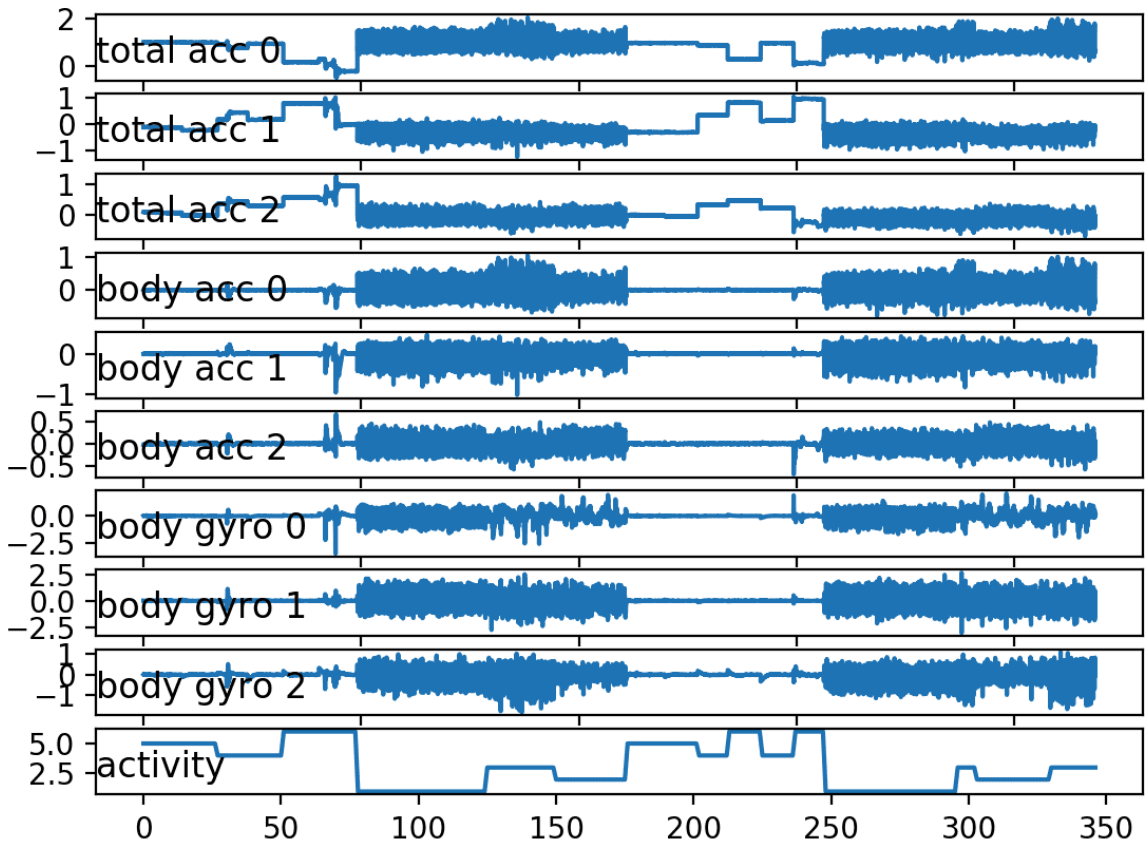
플롯에서 우리는 활동 1, 2, 3(걷기 활동)에 해당하는 큰 움직임 기간을 볼 수 있습니다. 또한 더 높은 번호의 활동 4, 5, 6(앉기, 서기, 누워 있기)에 대해 훨씬 적은 활동(즉, 비교적 직선)을 볼 수 있습니다.

이는 우리가 원시 데이터 세트를 올바르게 로드하고 해석했다는 좋은 확인입니다.

우리는 이 과목이 동일한 일반적인 활동 순서를 두 번 수행했고, 어떤 활동은 두 번 이상 수행되었음을 알 수 있습니다. 이는 주어진 과목에 대해 어떤 활동이 수행되었을지 또는 그 순서에 대해 가정해서는 안 된다는 것을 시사합니다.

우리는 또한 누워 있는 것과 같은 일부 정지 활동에 대해 비교적 큰 움직임을 볼 수 있습니다. 이것들이 이상치이거나 활동 전환과 관련이 있을 수 있습니다. 이러한 관찰 결과를 이상치로 매끄럽게 처리하거나 제거할 수 있습니다.

마지막으로, 우리는 9개 변수에서 많은 공통점을 봅니다. 예측 모델을 개발하는 데 이러한 추적의 하위 집합만 필요할 가능성이 매우 높습니다.



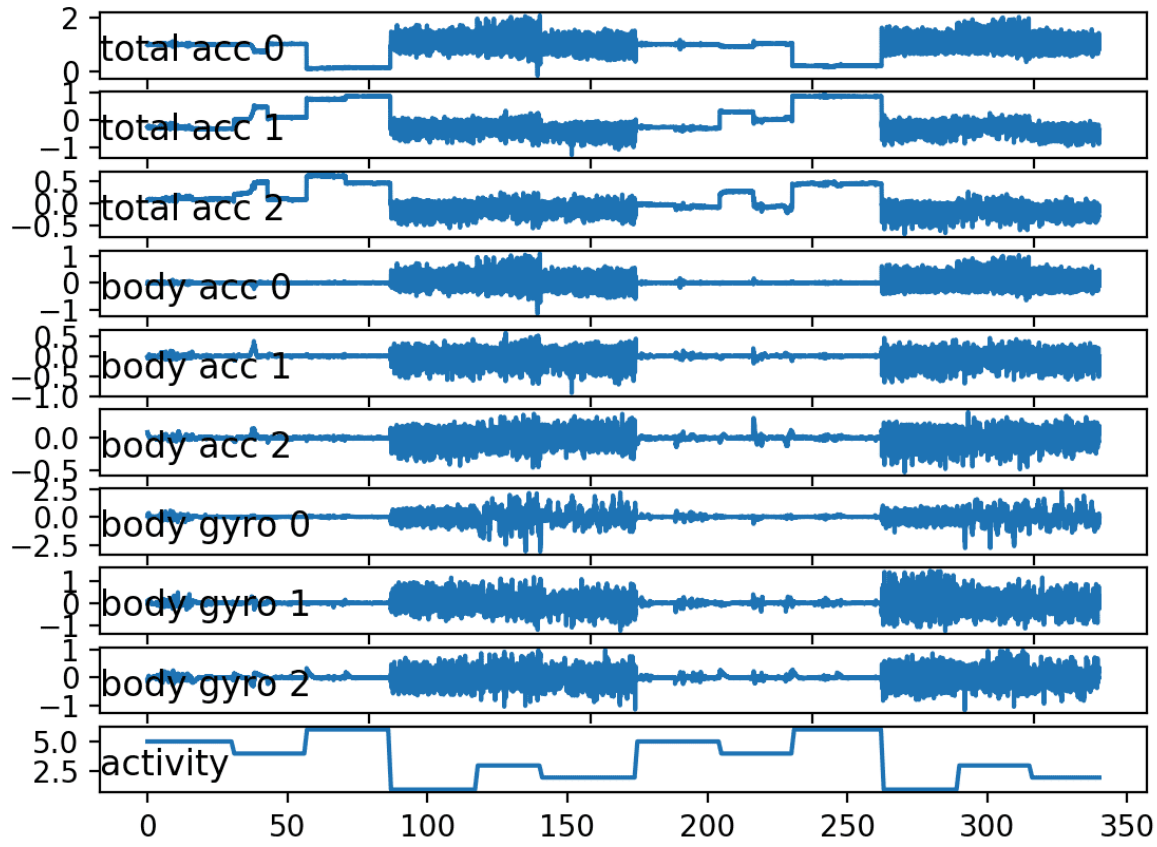
단일 주제에 대한 모든 변수에 대한 선형 플롯

훈련 데이터 세트에서 두 번째 주제의 식별자를 선택하는 것과 같이 작은 변경 하나만 하면 다른 주제에 대해 예제를 다시 실행할 수 있습니다.

```
# get the data for one subject
sub_id = train_subjects[1]
```

두 번째 주제의 줄거리도 비슷한 행동을 보이며 놀라운 점은 없습니다.

두 가지 활동의 연속은 첫 번째 주제보다 더 규칙적으로 보입니다.



두 번째 단일 주제에 대한 모든 변수에 대한 선형 플롯

7. 주제별 히스토그램 플롯

문제의 틀이 정해졌으므로, 우리는 일부 피험자의 움직임 데이터를 사용하여 다른 피험자의 움직임에서 발생하는 활동을 예측하는 데 관심을 두고 있습니다.

이는 피험자 간에 움직임 데이터에 규칙성이 있어야 함을 시사합니다. 데이터가 피험자별로 -1과 1 사이에서 조정되었다는 것을 알고 있으며, 이는 감지된 움직임의 진폭이 유사할 것임을 시사합니다.

또한 피험자들이 동일한 행동을 수행했다는 점에서 움직임 데이터의 분포도 피험자들 사이에서 유사할 것으로 예상합니다.

피험자 간에 운동 데이터의 히스토그램을 플로팅하고 비교하여 이를 확인할 수 있습니다. 유용한 접근 방식은 피험자당 하나의 플롯을 만들고 주어진 데이터의 세 축(예: 총 가속도)을 모두 플로팅한 다음 여러 피험자에 대해 이를 반복하는 것입니다. 플롯은 동일한 축을 사용하고 수평으로 정렬하여 피험자 간에 각 변수의 분포를 비교할 수 있도록 수정할 수 있습니다.

아래의 plot_subject_histograms() 함수는 이 동작을 구현합니다. 이 함수는 로드된 데이터 세트와 행을 피험자에 매핑하는 것, 그리고 기본적으로 10으로 고정된 최대 피험자 수를 가져옵니다.

각 주제에 대한 플롯이 생성되고 한 데이터 유형의 세 변수는 100개의 빈이 있는 히스토그램으로 플롯되어 분포를 명확하게 하는 데 도움이 됩니다. 각 플롯은 -1과 1의 경계에 고정된 동일한 축을 공유합니다.

```
# plot histograms for multiple subjects
def plot_subject_histograms(X, y, sub_map, n=10):
    pyplot.figure()
    # get unique subjects
    subject_ids = unique(sub_map[:,0])
    # enumerate subjects
    xaxis = None
    for k in range(n):
        sub_id = subject_ids[k]
        # get data for one subject
        subX, _ = data_for_subject(X, y, sub_map, sub_id)
        # total acc
        for i in range(3):
            ax = pyplot.subplot(n, 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(subX[:,:,i]), bins=100)
    pyplot.show()
```

전체 예는 아래와 같습니다.

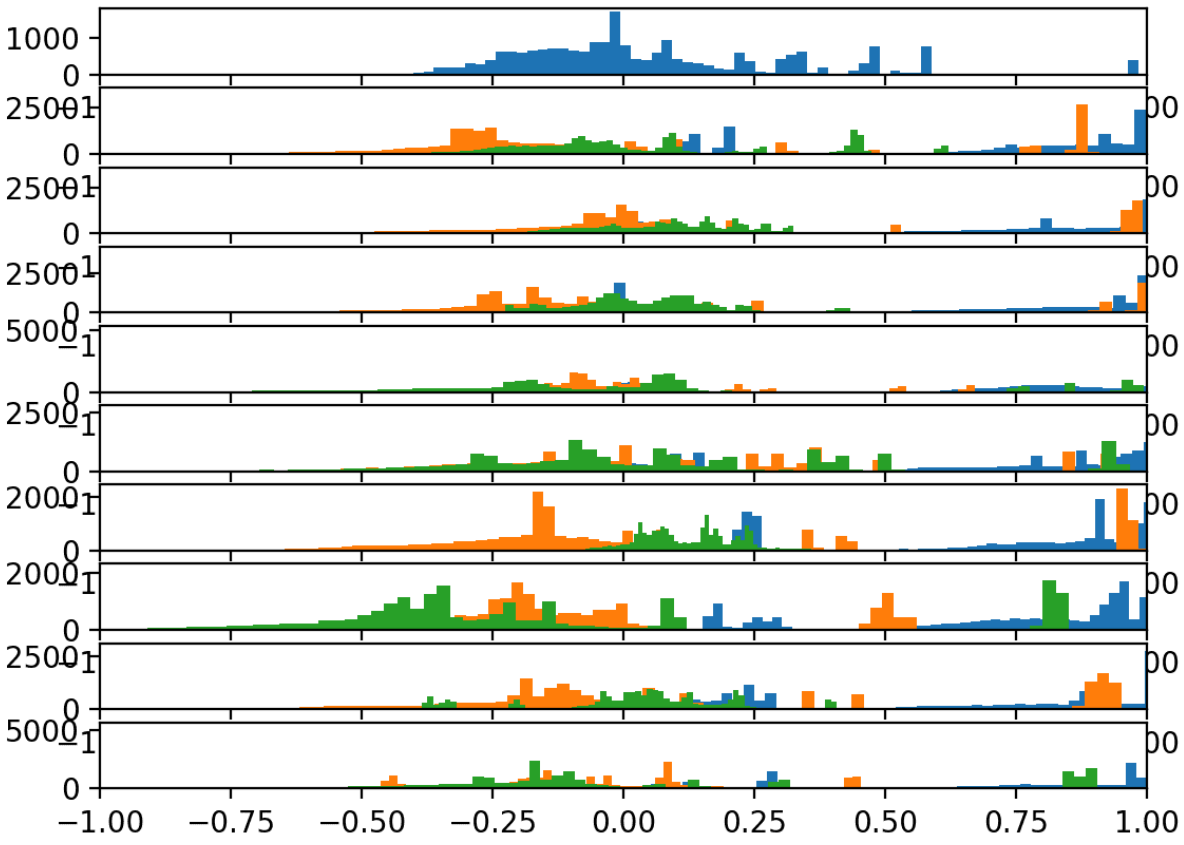
```
# plot histograms for multiple subjects
from numpy import array
from numpy import unique
from numpy import dstack
from pandas import read_csv
from matplotlib import pyplot

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded

# load a dataset group, such as train or test
def load_dataset(group, prefix=''):
    filepath = prefix + group + '/Inertial Signals/'
    # load all 9 files as a single array
    filenames = list()
    # total acceleration
    filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt', 'total_acc_z_'+group+'.txt']
    # body acceleration
    filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt', 'body_acc_z_'+group+'.txt']
    # body gyroscope
    filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt', 'body_gyro_z_'+group+'.txt']
    # load input data
    X = load_group(filenames, filepath)
    # load class output
    y = load_file(prefix + group + '/y_'+group+'.txt')
    return X, y

# get all data for one subject
def data_for_subject(X, y, sub_map, sub_id):
    # get row indexes for the subject id
    ix = [i for i in range(len(sub_map)) if sub_map[i]==sub_id]
    # return the selected samples
    return X[ix, :, :], y[ix]

# convert a series of windows to a 1D list
def to_series(windows):
    series = list()
    for window in windows:
        # remove the overlap from the window
        half = int(len(window) / 2) - 1
        for value in window[-half:]:
            series.append(value)
    return series

# plot histograms for multiple subjects
def plot_subject_histograms(X, y, sub_map, n=10):
    pyplot.figure()
    # get unique subjects
    subject_ids = unique(sub_map[:,0])
    # enumerate subjects
    xaxis = None
    for k in range(n):
        sub_id = subject_ids[k]
        # get data for one subject
        subX, _ = data_for_subject(X, y, sub_map, sub_id)
        # total acc
        for i in range(3):
            ax = pyplot.subplot(n, 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(subX[:,:,i]), bins=100)
    pyplot.show()

# load training dataset
X, y = load_dataset('train', 'HARDataset/')
# load mapping of rows to subjects
sub_map = load_file('HARDataset/train/subject_train.txt')
# plot histograms for subjects
plot_subject_histograms(X, y, sub_map)
```

예제를 실행하면 총 가속도 데이터의 3개 축에 대한 히스토그램이 있는 10개의 플롯이 있는 단일 그림이 생성됩니다.

주어진 플롯의 세 축은 각각 다른 색상을 갖습니다. 구체적으로 x, y, z는 각각 파란색, 주황색, 녹색입니다.

우리는 주어진 축에 대한 분포가 큰 개별 데이터 그룹에서 가우시안 분포로 나타나는 것을 볼 수 있습니다.

일부 분포가 일치하는 것을 볼 수 있습니다(예: 0.0을 중심으로 한 중앙에 주요 그룹이 있음). 이는 적어도 이 데이터의 경우 피험자 전체에서 움직임 데이터의 연속성이 어느 정도 있음을 시사합니다.



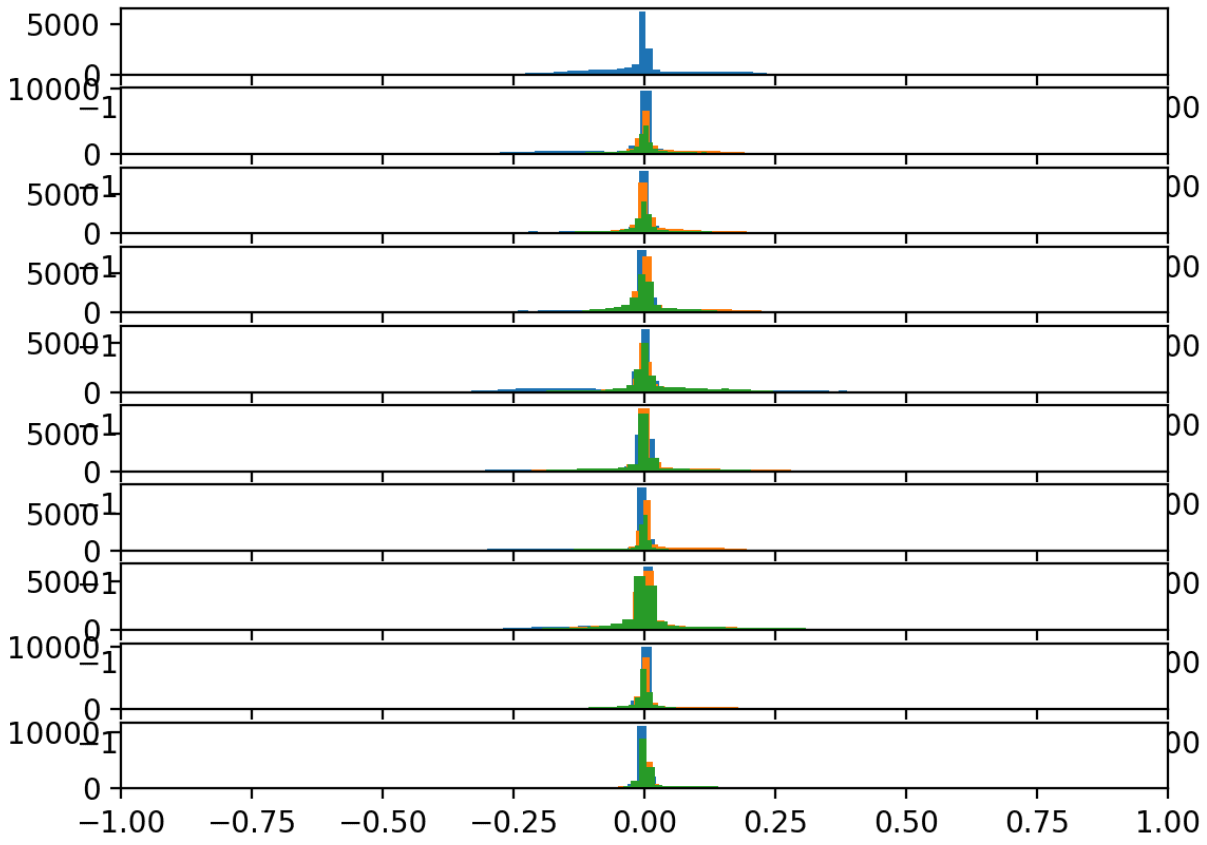
10명의 피험자에 대한 전체 가속도 데이터의 히스토그램

plot_subject_histograms() 함수를 업데이트하여 다음으로 신체 가속도 분포를 플롯할 수 있습니다. 업데이트된 함수는 아래에 나와 있습니다.

```
# plot histograms for multiple subjects
def plot_subject_histograms(X, y, sub_map, n=10):
    pyplot.figure()
    # get unique subjects
    subject_ids = unique(sub_map[:,0])
    # enumerate subjects
    xaxis = None
    for k in range(n):
        sub_id = subject_ids[k]
        # get data for one subject
        subX, _ = data_for_subject(X, y, sub_map, sub_id)
        # body acc
        for i in range(3):
            ax = pyplot.subplot(n, 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(subX[:,:,3+i]), bins=100)
    pyplot.show()
```

업데이트된 예제를 실행하면 매우 다른 결과를 보이는 동일한 플롯이 생성됩니다.

여기서 우리는 모든 데이터가 피험자 내 축과 피험자 간에 0.0을 중심으로 클러스터링되어 있는 것을 볼 수 있습니다. 이는 데이터가 중앙에 위치했을 수 있음을 시사합니다(평균 0). 피험자 간에 이러한 강력한 일관성은 모델링에 도움이 될 수 있으며, 전체 가속도 데이터에서 피험자 간 차이가 그다지 도움이 되지 않을 수 있음을 시사할 수 있습니다.



10명의 피험자에 대한 신체 가속도 데이터의 히스토그램

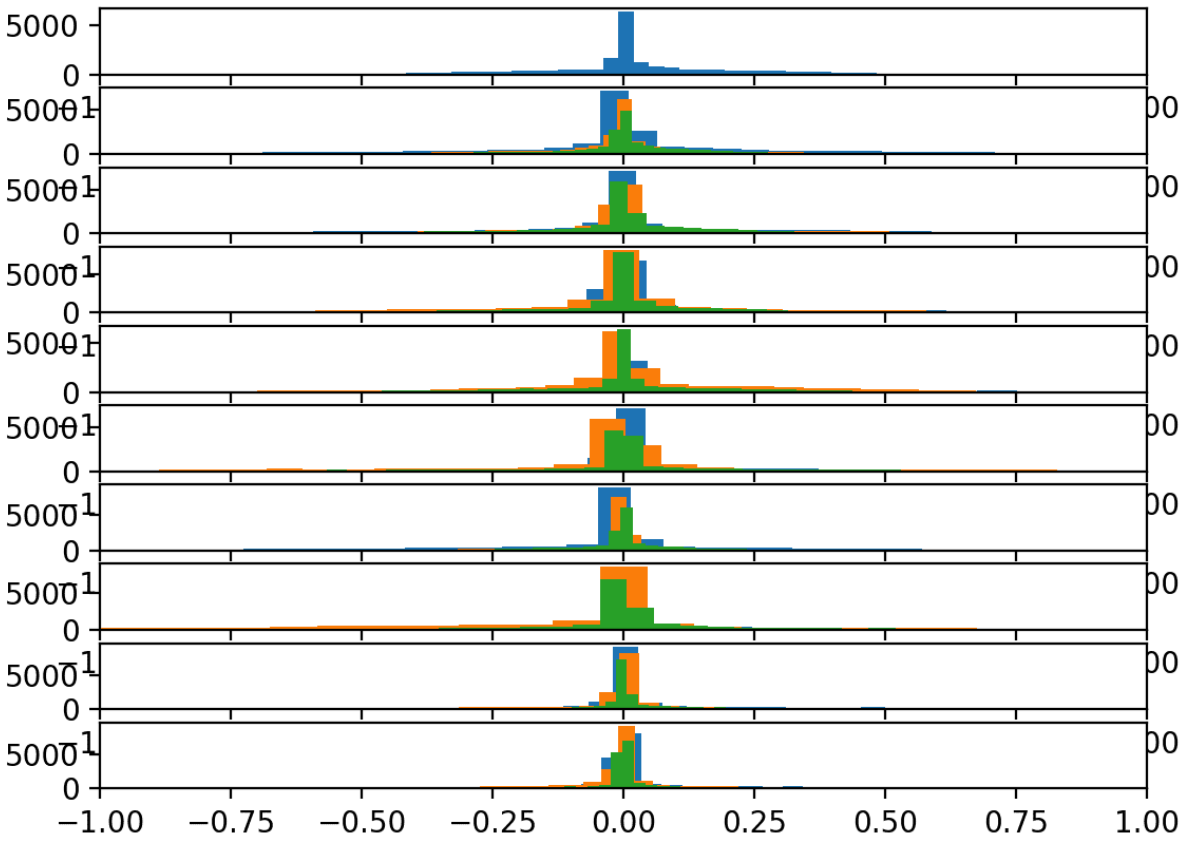
마지막으로 자이로스코프 데이터에 대한 최종 플롯을 생성할 수 있습니다.

업데이트된 기능은 아래와 같습니다.

```
# plot histograms for multiple subjects
def plot_subject_histograms(X, y, sub_map, n=10):
    pyplot.figure()
    # get unique subjects
    subject_ids = unique(sub_map[:,0])
    # enumerate subjects
    xaxis = None
    for k in range(n):
        sub_id = subject_ids[k]
        # get data for one subject
        subX, _ = data_for_subject(X, y, sub_map, sub_id)
        # body acc
        for i in range(3):
            ax = pyplot.subplot(n, 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(subX[:,:,6+i]), bins=100)
    pyplot.show()
```

예제를 실행하면 신체 가속도 데이터와 매우 유사한 결과가 나타납니다.

우리는 각 피험자에서 각 축에 대해 0.0을 중심으로 하는 가우시안 분포의 높은 가능성을 봅니다. 분포는 약간 더 넓고 꼬리가 더 굵지만, 이는 피험자 간의 움직임 데이터를 모델링하는 데 있어서 고무적인 발견입니다.



10명의 피험자에 대한 신체 자이로스코프 데이터의 히스토그램

8. 활동별 히스토그램 플롯

우리는 활동 데이터를 기반으로 활동을 구별하는 데 관심이 있습니다.

이에 대한 가장 간단한 사례는 단일 피험자의 활동을 구별하는 것입니다. 이를 조사하는 한 가지 방법은 활동별 피험자의 움직임 데이터 분포를 검토하는 것입니다. 단일 피험자의 다양한 활동에 대한 움직임 데이터 분포에 약간의 차이가 있을 것으로 예상합니다.

활동당 히스토그램 플롯을 만들고 각 플롯에 주어진 데이터 유형의 세 축을 배치하여 이를 검토할 수 있습니다. 다시 말해, 플롯을 수평으로 배열하여 활동별 각 데이터 축의 분포를 비교할 수 있습니다. 플롯을 따라 활동 간 분포에 차이가 있을 것으로 예상합니다.

먼저, 우리는 주체에 대한 추적을 활동별로 그룹화해야 합니다. 아래의 *data_by_activity()* 함수는 이 동작을 구현합니다.

```
# group data by activity
def data_by_activity(X, y, activities):
    # group windows by activity
    return {a:X[y[:,0]==a, :, :] for a in activities}
```

이제 우리는 주어진 주제에 대한 활동별 플롯을 만들 수 있습니다.

아래의 plot_activity_histograms *()* 함수는 주어진 주제에 대한 추적 데이터에 대해 이 함수를 구현합니다.

먼저, 데이터를 활동별로 그룹화한 다음, 각 활동에 대해 하나의 서브플롯을 만들고 데이터 유형의 각 축을 히스토그램으로 추가합니다. 이 함수는 데이터의 처음 세 가지 특징, 즉 총 가속도 변수만 열거합니다.

```
# plot histograms for each activity for a subject
def plot_activity_histograms(X, y):
    # get a list of unique activities for the subject
    activity_ids = unique(y[:,0])
    # group windows by activity
    grouped = data_by_activity(X, y, activity_ids)
    # plot per activity, histograms for each axis
    pyplot.figure()
    xaxis = None
    for k in range(len(activity_ids)):
        act_id = activity_ids[k]
        # total acceleration
        for i in range(3):
            ax = pyplot.subplot(len(activity_ids), 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
```

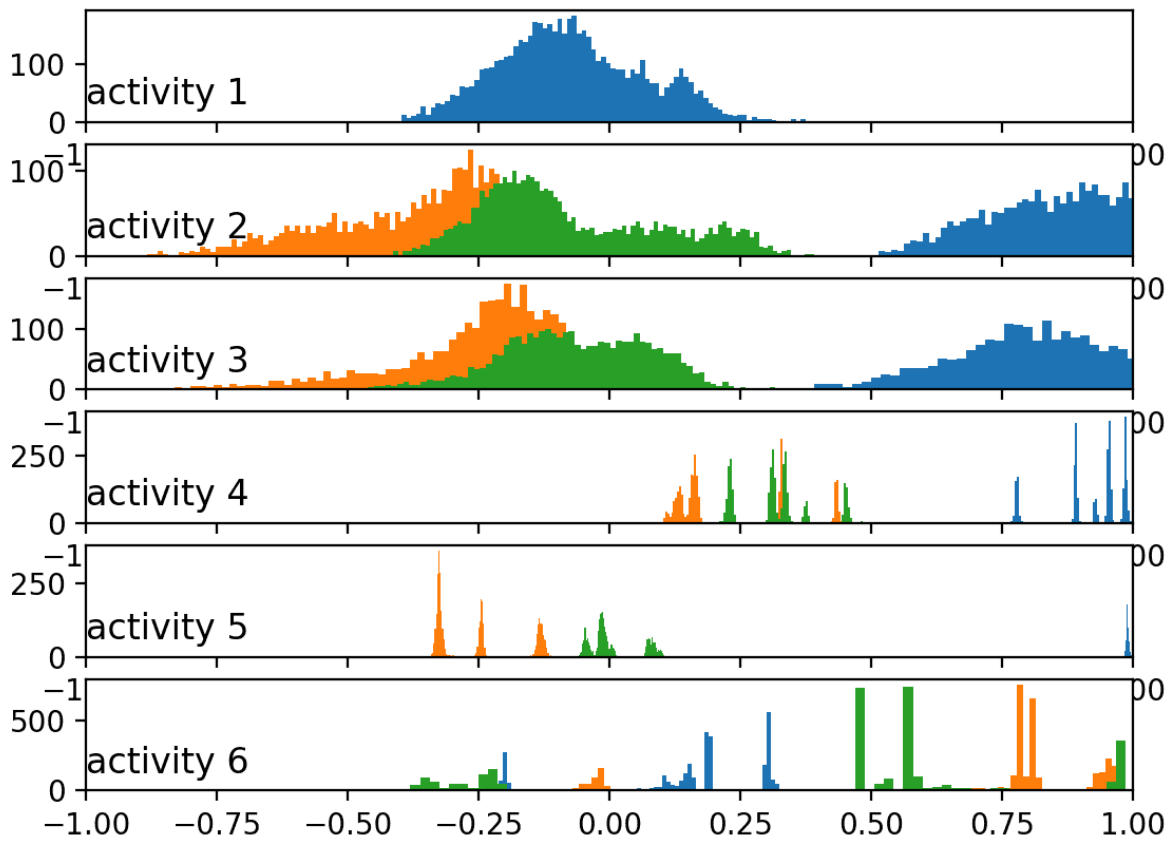
```
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(grouped[act_id][:,:,i]), bins=100)
            pyplot.title('activity '+str(act_id), y=0, loc='left')
    pyplot.show()
```

전체 예는 아래와 같습니다.

```
# plot histograms per activity for a subject
from numpy import array
from numpy import dstack
from numpy import unique
from pandas import read_csv
from matplotlib import pyplot

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded

# load a dataset group, such as train or test
def load_dataset(group, prefix=''):
    filepath = prefix + group + '/Inertial Signals/'
    # load all 9 files as a single array
    filenames = list()
    # total acceleration
    filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt', 'total_acc_z_'+group+'.txt']
    # body acceleration
    filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt', 'body_acc_z_'+group+'.txt']
    # body gyroscope
    filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt', 'body_gyro_z_'+group+'.txt']
    # load input data
    X = load_group(filenames, filepath)
    # load class output
    y = load_file(prefix + group + '/y_'+group+'.txt')
    return X, y

# get all data for one subject
def data_for_subject(X, y, sub_map, sub_id):
    # get row indexes for the subject id
    ix = [i for i in range(len(sub_map)) if sub_map[i]==sub_id]
    # return the selected samples
    return X[ix, :, :], y[ix]

# convert a series of windows to a 1D list
def to_series(windows):
    series = list()
    for window in windows:
        # remove the overlap from the window
        half = int(len(window) / 2) - 1
        for value in window[-half:]:
            series.append(value)
    return series

# group data by activity
def data_by_activity(X, y, activities):
    # group windows by activity
    return {a:X[y[:,0]==a, :, :] for a in activities}

# plot histograms for each activity for a subject
def plot_activity_histograms(X, y):
    # get a list of unique activities for the subject
    activity_ids = unique(y[:,0])
    # group windows by activity
    grouped = data_by_activity(X, y, activity_ids)
    # plot per activity, histograms for each axis
    pyplot.figure()
    xaxis = None
    for k in range(len(activity_ids)):
        act_id = activity_ids[k]
        # total acceleration
        for i in range(3):
            ax = pyplot.subplot(len(activity_ids), 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(grouped[act_id][:,:,i]), bins=100)
            pyplot.title('activity '+str(act_id), y=0, loc='left')
    pyplot.show()

# load data
trainX, trainy = load_dataset('train', 'HARDataset/')
# load mapping of rows to subjects
sub_map = load_file('HARDataset/train/subject_train.txt')
train_subjects = unique(sub_map)
# get the data for one subject
sub_id = train_subjects[0]
subX, suby = data_for_subject(trainX, trainy, sub_map, sub_id)
# plot data for subject
plot_activity_histograms(subX, suby)
```

예제를 실행하면 6개의 서브플롯이 있는 플롯이 생성되는데, 이는 트레인 데이터 세트의 첫 번째 피험자에 대한 각 활동에 대한 것입니다. 총 가속도 데이터의 각 x, y, z 축은 각각 파란색, 주황색, 녹색 히스토그램을 갖습니다.

각 활동이 다른 데이터 분포를 가지고 있으며, 큰 움직임(처음 세 활동)과 정지 활동(마지막 세 활동) 사이에 현저한 차이가 있음을 알 수 있습니다. 처음 세 활동의 데이터 분포는 평균과 표준 편차가 다를 수 있는 가우시안 분포처럼 보입니다. 후자의 활동의 분포는 다중 모드(즉, 여러 피크)처럼 보입니다.



활동별 총 가속도 데이터의 히스토그램

대신, 신체 가속도 데이터를 표시하는 *plot_activity_histograms()* 의 업데이트된 버전을 사용하여 동일한 예제를 다시 실행할 수 있습니다 .

업데이트된 기능은 아래와 같습니다.

```
# plot histograms for each activity for a subject
def plot_activity_histograms(X, y):
    # get a list of unique activities for the subject
    activity_ids = unique(y[:,0])
    # group windows by activity
    grouped = data_by_activity(X, y, activity_ids)
    # plot per activity, histograms for each axis
    pyplot.figure()
    xaxis = None
    for k in range(len(activity_ids)):
        act_id = activity_ids[k]
        # total acceleration
        for i in range(3):
            ax = pyplot.subplot(len(activity_ids), 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(grouped[act_id][:,:,3+i]), bins=100)
            pyplot.title('activity '+str(act_id), y=0, loc='left')
    pyplot.show()
```

업데이트된 예제를 실행하면 새로운 플롯이 생성됩니다.

여기서 우리는 움직이는 활동과 정지한 활동 사이에서 더 유사한 분포를 볼 수 있습니다. 데이터는 움직이는 활동의 경우 이중 모드처럼 보이고 정지한 활동의 경우 가우시안 또는 지수처럼 보입니다.

활동별 총 가속도 대 신체 가속도 분포에서 보이는 패턴은 이전 섹션에서 피험자 간에 동일한 데이터 유형에서 보이는 패턴을 반영합니다. 아마도 총 가속도 데이터가 활동을 구별하는 데 핵심일 것입니다.

활동별 신체 가속도 데이터의 히스토그램

마지막으로, 자이로스코프 데이터에 대한 활동별 히스토그램을 그리는 예제를 한 번 더 업데이트할 수 있습니다.

업데이트된 기능은 아래와 같습니다.

```
# plot histograms for each activity for a subject
def plot_activity_histograms(X, y):
    # get a list of unique activities for the subject
    activity_ids = unique(y[:,0])
    # group windows by activity
    grouped = data_by_activity(X, y, activity_ids)
    # plot per activity, histograms for each axis
    pyplot.figure()
    xaxis = None
    for k in range(len(activity_ids)):
        act_id = activity_ids[k]
        # total acceleration
        for i in range(3):
            ax = pyplot.subplot(len(activity_ids), 1, k+1, sharex=xaxis)
            ax.set_xlim(-1,1)
            if k == 0:
                xaxis = ax
            pyplot.hist(to_series(grouped[act_id][:,:,6+i]), bins=100)
            pyplot.title('activity '+str(act_id), y=0, loc='left')
    pyplot.show()
```

예제를 실행하면 신체 가속도 데이터와 유사한 패턴을 갖는 플롯이 생성됩니다. 다만 움직이는 활동에 대한 이중 모드 분포 대신 꼬리가 두꺼운 가우시안 유사 분포가 나타날 수 있습니다.

활동별 신체 자이로스코프 데이터의 히스토그램

이 모든 플롯은 첫 번째 주제를 위해 만들어졌고, 다른 주제의 활동 전반에 걸친 움직임 데이터의 유사한 분포와 관계가 나타날 것으로 예상합니다.

9. 플롯 활동 기간 상자 그림

고려해야 할 마지막 측면은 피험자가 각 활동에 얼마나 많은 시간을 할애하는가입니다.

이는 클래스의 균형과 밀접한 관련이 있습니다. 활동(클래스)이 데이터 세트 내에서 일반적으로 균형을 이루고 있다면, 추적 과정에서 주어진 주제에 대한 활동의 균형도 상당히 잘 균형을 이룰 것으로 예상합니다.

이를 확인하려면 각 피험자가 각 활동에 소요한 시간(표본 또는 행 단위)을 계산하고 각 활동에 대한 기간 분포를 살펴보면 됩니다.

이 데이터를 검토하는 편리한 방법은 중앙값(선), 중간 50%(상자), 사분위 범위(수염)로 데이터의 전반적인 범위, 그리고 이상치(점)를 보여주는 상자 그림으로 분포를 요약하는 것입니다.

아래의 plot_activity_durations_by_subject() 함수는 먼저 주제별로 데이터 세트를 분할한 다음, 주제별 데이터를 활동별로 분할하고 각 활동에 소요된 행을 세는 방식으로 이 동작을 구현합니다. 마지막으로 활동별 기간 측정값의 상자 그림을 만듭니다.

```
# plot activity durations by subject
def plot_activity_durations_by_subject(X, y, sub_map):
    # get unique subjects and activities
    subject_ids = unique(sub_map[:,0])
    activity_ids = unique(y[:,0])
    # enumerate subjects
    activity_windows = {a:list() for a in activity_ids}
    for sub_id in subject_ids:
        # get data for one subject
        _, subj_y = data_for_subject(X, y, sub_map, sub_id)
        # count windows by activity
        for a in activity_ids:
            activity_windows[a].append(len(subj_y[subj_y[:,0]==a]))
    # organize durations into a list of lists
    durations = [activity_windows[a] for a in activity_ids]
    pyplot.boxplot(durations, labels=activity_ids)
    pyplot.show()
```

전체 예는 아래와 같습니다.

```
# plot durations of each activity by subject
from numpy import array
```

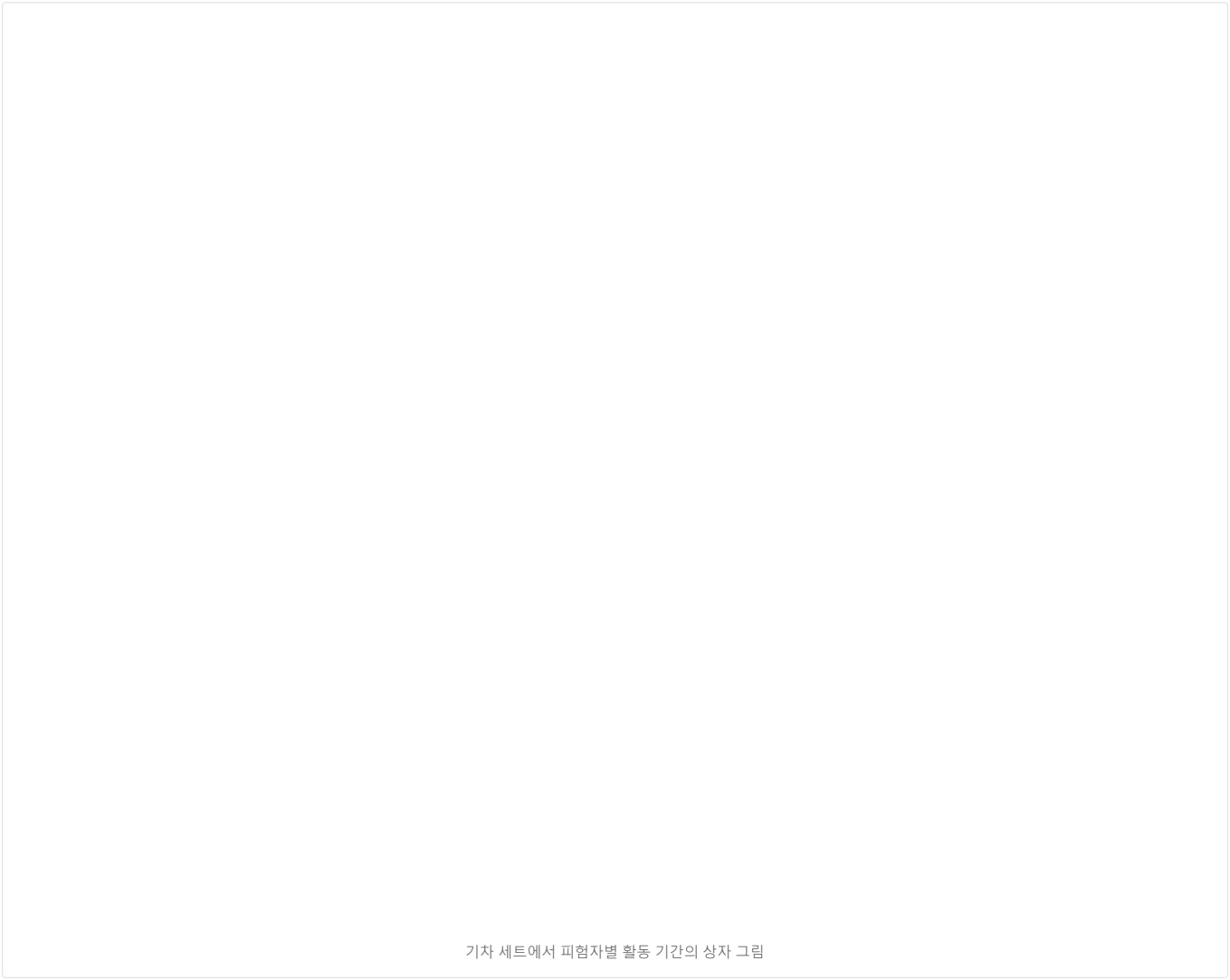
```
from numpy import dstack
from numpy import unique
from pandas import read_csv
from matplotlib import pyplot

# load a single file as a numpy array
def load_file(filepath):
    dataframe = read_csv(filepath, header=None, delim_whitespace=True)
    return dataframe.values

# load a list of files, such as x, y, z data for a given variable
def load_group(filenames, prefix=''):
    loaded = list()
    for name in filenames:
        data = load_file(prefix + name)
        loaded.append(data)
    # stack group so that features are the 3rd dimension
    loaded = dstack(loaded)
    return loaded

# load a dataset group, such as train or test
def load_dataset(group, prefix=''):
    filepath = prefix + group + '/Inertial Signals/'
    # load all 9 files as a single array
    filenames = list()
    # total acceleration
    filenames += ['total_acc_x_'+group+'.txt', 'total_acc_y_'+group+'.txt', 'total_acc_z_'+group+'.txt']
    # body acceleration
    filenames += ['body_acc_x_'+group+'.txt', 'body_acc_y_'+group+'.txt', 'body_acc_z_'+group+'.txt']
    # body gyroscope
    filenames += ['body_gyro_x_'+group+'.txt', 'body_gyro_y_'+group+'.txt', 'body_gyro_z_'+group+'.txt']
    # load input data
    X = load_group(filenames, filepath)
    # load class output
    y = load_file(prefix + group + '/y_'+group+'.txt')
    return X, y

# get all data for one subject
def data_for_subject(X, y, sub_map, sub_id):
    # get row indexes for the subject id
    ix = [i for i in range(len(sub_map)) if sub_map[i]==sub_id]
    # return the selected samples
    return X[ix, :, :], y[ix]

# convert a series of windows to a 1D list
def to_series(windows):
    series = list()
    for window in windows:
        # remove the overlap from the window
        half = int(len(window) / 2) - 1
        for value in window[-half:]:
            series.append(value)
    return series

# group data by activity
def data_by_activity(X, y, activities):
    # group windows by activity
    return {a:X[y[:,0]==a, :, :] for a in activities}

# plot activity durations by subject
def plot_activity_durations_by_subject(X, y, sub_map):
    # get unique subjects and activities
    subject_ids = unique(sub_map[:,0])
    activity_ids = unique(y[:,0])
    # enumerate subjects
    activity_windows = {a:list() for a in activity_ids}
    for sub_id in subject_ids:
        # get data for one subject
        _, subj_y = data_for_subject(X, y, sub_map, sub_id)
        # count windows by activity
        for a in activity_ids:
            activity_windows[a].append(len(subj_y[subj_y[:,0]==a]))
    # organize durations into a list of lists
    durations = [activity_windows[a] for a in activity_ids]
    pyplot.boxplot(durations, labels=activity_ids)
    pyplot.show()

# load training dataset
X, y = load_dataset('train', 'HARDataset/')
# load mapping of rows to subjects
sub_map = load_file('HARDataset/train/subject_train.txt')
# plot durations
plot_activity_durations_by_subject(X, y, sub_map)
```

이 예제를 실행하면 각 활동별로 총 6개의 상자 그림이 생성됩니다.

각 상자 그림은 훈련 데이터 세트에 있는 피험자들이 각 활동에 소요한 시간(행 또는 창 개수)을 요약합니다.

피험자들은 정지 활동(4, 5, 6)에 더 많은 시간을 보냈고, 움직이는 활동(1, 2, 3)에는 더 적은 시간을 보냈으며, 3의 분포가 가장 작거나 시간이 가장 적게 소요된 것을 알 수 있습니다.

활동 전체에 걸친 확산이 크지 않아 더 긴 기간의 활동을 다듬거나 진행 중인 활동을 과도하게 샘플링할 필요성이 거의 없음을 시사합니다. 그러나 이러한 접근 방식은 진행 중인 활동에 대한 예측 모델의 기술이 일반적으로 더 나쁠 경우 여전히 사용할 수 있습니다.



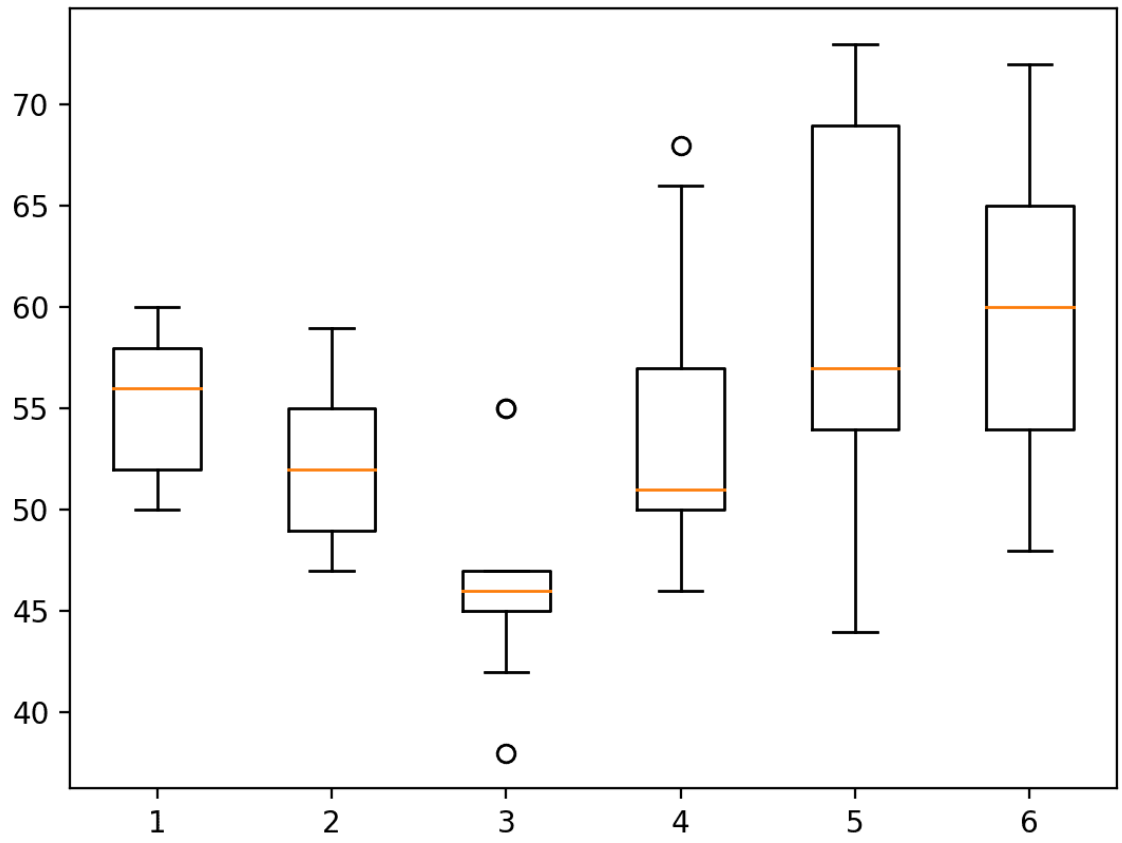
기차 세트에서 피험자별 활동 기간의 상자 그림

다음과 같은 추가 줄을 사용하여 훈련 데이터에 대해 비슷한 상자 그림을 만들 수 있습니다.

```
# load test dataset
X, y = load_dataset('test', 'HARDataset/')
# load mapping of rows to subjects
sub_map = load_file('HARDataset/test/subject_test.txt')
# plot durations
plot_activity_durations_by_subject(X, y, sub_map)
```

업데이트된 예제를 실행하면 활동 간에 유사한 관계가 있음을 알 수 있습니다.

이는 테스트 및 훈련 데이터 세트가 전체 데이터 세트를 상당히 대표한다는 것을 보여주기 때문에 고무적입니다.



테스트 세트의 피험자별 활동 기간의 상자 그림

이제 데이터 세트를 살펴보았으니 이를 모델링하는 방법에 대한 몇 가지 아이디어를 제안할 수 있습니다.

10. 모델링 접근 방식

이 섹션에서는 활동 인식 데이터 세트를 모델링하는 몇 가지 접근 방식을 요약합니다.

이러한 아이디어는 프로젝트의 주요 주제로 나뉩니다.

문제 프레이밍

첫 번째로 중요한 고려 사항은 예측 문제의 프레이밍입니다.

원래 연구 내용에서 설명한 문제의 구성은 기존 피험자의 움직임 데이터와 활동을 바탕으로, 새로운 피험자의 움직임 데이터를 바탕으로 해당 피험자의 활동을 예측하는 것입니다.

이를 요약하면 다음과 같습니다.

- 이동 데이터 창을 통해 활동을 예측합니다.

이는 문제에 대한 합리적이고 유용한 설명입니다.

제공된 데이터를 예측 문제로 표현할 수 있는 다른 가능한 방법은 다음과 같습니다.

- 이동 데이터의 시간 단계를 고려하여 활동을 예측합니다.
- 여러 창의 이동 데이터를 바탕으로 활동을 예측합니다.
- 여러 창의 이동 데이터를 바탕으로 활동 순서를 예측합니다.
- 미리 분할된 활동에 대한 일련의 움직임 데이터를 바탕으로 활동을 예측합니다.
- 이동 데이터의 시간 단계를 고려하여 활동 중단이나 전환을 예측합니다.
- 이동 데이터 창이 주어진 경우 정지 또는 비정지 활동을 예측합니다.

이러한 프레이밍 중 일부는 너무 어렵거나 너무 쉬울 수 있습니다.

그럼에도 불구하고, 이러한 프레이밍은 데이터 세트를 탐색하고 이해할 수 있는 추가적인 방법을 제공합니다.

데이터 준비

원시 데이터를 사용하여 모델을 학습시키기 전에 일부 데이터 준비가 필요할 수 있습니다.

데이터는 이미 [-1,1] 범위로 조정된 것으로 보입니다.

모델링에 앞서 수행할 수 있는 추가 데이터 변환은 다음과 같습니다.

- 주제 간 정규화.
- 과목별 표준화.
- 과목 간 표준화.
- 축 기능 선택.
- 데이터 유형 기능 선택.
- 신호 이상치 감지 및 제거.
- 과도하게 표현된 활동의 창을 제거합니다.
- 대표성이 부족한 활동에 대한 과다 샘플링 창.
- 신호 데이터를 섹션의 1/4, 1/2, 1, 2 또는 다른 분수로 다운샘플링합니다.

예측 모델링

일반적으로 이 문제는 시계열 다중 클래스 분류 문제입니다.

앞서 살펴본 것처럼 이는 이진 분류 문제와 다단계 시계열 분류 문제로도 해석될 수 있습니다.

원래 논문은 각 데이터 창에서 피처가 엔지니어링된 데이터 세트 버전에서 고전적인 머신 러닝 알고리즘의 사용을 탐구했습니다. 구체적으로는 수정된 지원 벡터 머신입니다.

데이터 세트의 특성 공학적 버전에 대한 SVM의 결과는 문제에 대한 성능의 기준을 제공할 수 있습니다.

이 지점에서 확장하여, 이 버전의 데이터 세트에서 여러 개의 선형, 비선형 및 앙상블 머신 러닝 알고리즘을 평가하면 더 나은 벤치마크를 제공할 수 있습니다.

문제의 초점은 엔지니어링되지 않은 버전이나 원시 버전의 데이터 세트에 맞춰질 수 있습니다.

여기서는 문제에 가장 적합한 모델을 결정하기 위해 모델 복잡도의 진행 과정을 살펴볼 수 있습니다. 살펴볼 수 있는 후보 모델은 다음과 같습니다.

- 일반적인 선형, 비선형 및 앙상블 머신 러닝 알고리즘입니다.
- 다층 퍼셉트론.
- 합성곱 신경망, 특히 1D CNN.
- 순환 신경망, 특히 LSTM.
- CNN과 LSTMS의 하이브리드로는 CNN-LSTM, ConvLSTM 등이 있습니다.

모델 평가

원래 논문에서는 모델을 평가하기 위해 피험자별로 데이터를 70% 대 30% 비율로 분할하여 훈련/테스트를 진행했습니다.

미리 정의된 데이터 분할을 살펴보면 두 세트 모두 전체 데이터 세트를 비교적 잘 대표한다는 것을 알 수 있습니다.

또 다른 대안적 방법은 피험자당 leave-one-out 교차 검증 또는 LOOCV를 사용하는 것입니다. 각 피험자의 데이터를 보류 테스트 세트로 사용할 수 있는 기회를 제공하는 것 외에도 이 접근 방식은 평균을 내고 요약할 수 있는 30개 점수의 모집단을 제공하여 보다 견고한 결과를 제공할 수 있습니다.

모델 성능은 분류 정확도와 혼동 행렬을 사용하여 표현되었는데, 둘 다 예측 문제의 다중 클래스 특성에 적합합니다.

구체적으로, 혼동 행렬은 어떤 클래스가 다른 클래스보다 예측하기 쉬운지 또는 예측하기 어려운지 판단하는 데 도움이 됩니다. 예를 들어, 정지 활동과 운동을 포함하는 활동의 클래스가 그렇습니다.

추가 읽기

이 섹션에서는 해당 주제에 대해 더 자세히 알아보고 싶은 경우에 대비해 더 많은 자료를 제공합니다.

서류

- 센서 기반 활동 인식을 위한 딥러닝: 조사 .
- 스마트폰을 이용한 인간 활동 인식을 위한 퍼블릭 도메인 데이터세트 , 2013년.
- 2012년 , 다중클래스 하드웨어 친화적 지원 벡터 머신을 사용한 스마트폰에서의 인간 활동 인식 .

API

- pandas.read_csv API
- numpy.dstack API

조항

- 스마트폰 데이터 세트를 사용한 인간 활동 인식, UCI 머신 러닝 리포지토리
- 활동 인식, 위키피디아
- 스마트폰 센서를 활용한 활동 인식 실험 , 비디오.

요약

이 튜토리얼에서는 시계열 분류를 위한 스마트폰을 활용한 활동 인식 데이터세트를 알아보고, 예측 모델링에 적합하도록 데이터세트를 로드하고 탐색하는 방법을 알아보았습니다.

구체적으로 다음 내용을 배웠습니다.

- 데이터 세트를 메모리에 다운로드하고 로드하는 방법.
- 선형 플롯, 히스토그램, 상자 그림을 사용하여 동작 데이터의 구조를 더 잘 이해하는 방법.
- 프레이밍, 데이터 준비, 모델링, 평가를 포함한 문제를 모델링하는 방법입니다.

질문이 있으신가요?
아래의 댓글에 질문을 남겨주세요. 최선을 다해 답변해드리겠습니다.

이 주제에 대한 추가 정보

표준 인간 활동에 대한 간단한 소개...

인간을 위한 머신 러닝 알고리즘 평가...

인간을 위한 1D 합성 신경망 모델...

인간 활동 인식을 위한 LSTM 시계열...

인간 활동 인식을 위한 딥 러닝 모델

활동을 통해 일반화 오류를 줄이는 방법...

제이슨 브라운리 소개

제이슨 브라운리 박사는 개발자에게 실습 튜토리얼을 통해 최신 머신 러닝 방법을 활용해 결과를 얻는 방법을 가르치는 머신 러닝 전문가입니다.
Jason Brownlee의 모든 게시물 보기 →

‹파이썬에서 머신 러닝 알고리즘을 스팟 체크하는 프레임워크를 개발하는 방법 인간 활동 인식을 위한 머신 러닝 알고리즘 평가›

스마트폰 데이터에서 인간 활동을 모델링하는 방법 에 대한 92가지 응답

Sid 2018년 9월 17일 오후 1시 14분 #

회신하다

제이슨, 언제나처럼 평화롭게 일하시네요. 다만 샘플링된 데이터를 로딩하는 대신 전화(예를 들어 안드로이드)와 통신할 수 있는 코드가 있었으면 좋았을 텐데요.

Jason Brownlee 2018년 9월 17일 오후 2시 8분 #

회신하다

감사해요.

좋은 제안이에요!

Noam Cohen 2018년 9월 21일 오후 6시 45분 #

회신하다

휴대전화에서 데이터를 수집하는 방법을 보여주는 오픈 소스가 여러 개 있습니다(github 참조).
제 앱(개발 중)에서 가속 데이터를 수집하고(10초 슬라이딩 윈도우 사용) 특정 트리거 시 임시 파일에 저장하고 Google 클라우드에 업로드합니다. 성가신 보안 때문에 쉬운 작업은 아닙니다.
누군가 관심이 있다면 알려주세요. 샘플 앱을 만들어보도록 하겠습니다.

Jason Brownlee 2018년 9월 22일 오전 6시 27분 #

회신하다

도전적인 문제인 것 같네요!

Shishir Pandey 2018년 10월 3일 오전 3:51 #

회신하다

응용 프로그램을 통해 활동을 인식하는 방법을 알려주세요

Emmanuel Awotunde 2019년 3월 20일 오전 4:32 #

회신하다

저는 이 분야에 매우 관심이 있습니다. 저는 6개 단원 과정인 학교 프로젝트를 진행하고 있습니다. 저는 인간 활동 인식에 대해 읽고 주제로 선택했습니다. 모바일 애플리케이션 개발도 시작했습니다. 필요한 슬라이딩 윈도우로 가속도계와 자이로스코프 데이터를 수집하는 방법과 이 데이터를 예측에 사용하는 방법에 대해 막혔습니다.

Jason Brownlee 2019년 3월 20일 오전 8시 35분 #

회신하다

먼저 전화로 데이터를 수집한 다음 필요한 경우 슬라이딩 윈도우 방식으로 재구성하는 것이 좋을 것 같습니다.

Udathhu Manasa 2019년 12월 20일 오후 4:24 #

회신하다

안녕하세요 선생님
저는 모바일 폰에서 데이터를 가져와서 안드로이드 애플리케이션(인간 활동 인식)을 개발하는 데 관심이 있습니다.
데이터를 가져오는 방법을 알려주세요.

Jason Brownlee 2019년 12월 21일 오전 7시 6분 #

회신하다

죄송합니다. 안드로이드에 대해서는 잘 모르겠어요.

Rebeen 2018년 9월 17일 오후 2시 57분 #

회신하다

매우 감사합니다 ,

활동 인식이나 활동에서의 이상 감지에 관한 기사가 있다면 알려 주시기 바랍니다. 다시 한번 감사드립니다.

Jason Brownlee 2018년 9월 18일 오전 6시 10분 #

회신하다

저는 활동 인식에 관한 여러 게시물을 올릴 예정입니다.

이 책에서도 이 주제에 대해 자세히 다루고 있습니다:
https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/

AEPL 2018년 9월 17일 오후 8시 18분 #

회신하다

공유해 주셔서 감사합니다.

Jason Brownlee 2018년 9월 18일 오전 6시 11분 #

회신하다

천만에요.

Samih Eisa 2018년 9월 21일 오전 8:11 #

회신하다

고맙습니다, 제이슨.

Jason Brownlee 2018년 9월 21일 오전 10시 55분 #

회신하다

천만에요.

Noam Cohen 2018년 9월 21일 오후 6시 39분 #

회신하다

비슷한 종류의 낙상에 대한 데이터 세트가 있나요? 저는 노인의 낙상 감지에 대한 많은 참고 자료를 찾았지만, 젊은 사람(20~60세)의 낙상 감지에 대한 참고 자료는 찾지 못했습니다. 특히, 라이딩 중 낙상을 식별하고 싶습니다(속도 데이터가 없기 때문에 예상보다 쉽지 않습니다).

블로그 게시물에 감사드립니다!

Jason Brownlee 2018년 9월 22일 오전 6시 27분 #

회신하다

있을 것 같지만 잘 모르겠어요. 검색해 보는 게 어떨까요?

servomac 2018년 9월 21일 오후 6시 54분 #

회신하다

저는 이 같은 문제에 대해 LSTM을 모델링하기 위해 Keras를 사용하여 모델을 구현했습니다. https://github.com/servomac/Human-Activity-Recognition 에서 살펴보실 수 있습니다 .

Jason Brownlee 2018년 9월 22일 오전 6시 27분 #

회신하다

좋은 일이에요!

Aman Ullah 2018년 9월 23일 오전 2시 13분 #

회신하다

안녕

질문이 있습니다. 만약 연구가 실제 데이터에 관심이 있다면 연구자는 왜 데이터를 변환합니까? 변환에는 다른 패턴과 모델이 있습니다.

Jason Brownlee 2018년 9월 23일 오전 6시 40분 #

회신하다

네, 우리는 데이터 패턴의 구조를 학습 알고리즘에 더 잘 노출시키기 위해 데이터를 변환합니다. 다른 이유는 없습니다.

John Wingate 2018년 9월 24일 오전 7시 40분 #

회신하다

인간 활동 인식에 관한 훌륭한 게시물입니다.

Jason Brownlee 2018년 9월 24일 오전 8시 37분 #

회신하다

고맙습니다, 존!

Shishir Pandey 2018년 10월 9일 오전 5:35 #

회신하다

스마트폰으로 실시간으로 사람의 활동을 인식하고 싶은데, 도와주세요.

Jason Brownlee 2018년 10월 9일 오전 8시 46분 #

회신하다

죄송합니다. 스마트폰에서 실행되는 예제는 없습니다.

Joe 2018년 12월 2일 오전 2시 42분 #

회신하다

안녕하세요!
저는 Wireless Sensor Data Mining(WISDM) 데이터 세트를 사용하고 있으며, 여기에는 다음이 포함됩니다:
http://www.cis.fordham.edu/wisdm/dataset.php
예제 수: 1,098,207
속성 수: 6
클래스 분포
걷기: 424,400(38.6%)
조깅: 342,177(31.2%)
위층: 122,869(11.2%)
아래층: 100,427(9.1%)
앉아 있기: 59,939(5.5%)
서 있기: 48,395(4.4%)
제 질문은 이 데이터 세트의 데이터 준비에 관한 것입니다.
저는 CONV2D-LSTM 네트워크를 사용하려고 하므로 90×3 크기를 선택했습니다.
제가 하는 방법은 45 스텝 오버 데이터가 있는 90×3 창을 수직으로 옮기는 것입니다.
우선 이 겹침(45)이 도움이 되나요?
그런 다음 60%의 데이터를 훈련에, 20%를 테스트에, 20%를 검증에 선택합니다. 방법은 각 사람의 데이터에서 60%, 20%, 20%를 선택하여 각 사람(36)이 훈련, 테스트, 검증에서 각자의 몫을 가지고 있는지 확인하는 것입니다.
제 아이디어는 괜찮은가요?
감사합니다 .

Jason Brownlee 2018년 12월 2일 오전 6시 22분 #

회신하다

죄송합니다. 이 데이터 세트에 익숙하지 않습니다. 실험하고 선택한 구성이 안정적인지 확인하는 것이 좋습니다.

Joe 2018년 12월 2일 오전 8시 50분 #

회신하다

고맙습니다.
제 석사 논문인데, 당신의 답변이 좋은 지적이라고 생각합니다.
감사합니다.
그리고 전문가로서, 이런 종류의 데이터와 활동에 적합한 레이어 수는 어떻게 생각하십니까? 저는 CONV2D(특징 추출기)와 LSTM을 사용하고 있습니다.

Jason Brownlee 2018년 12월 3일 오전 6시 37분 #

회신하다

특정 데이터 세트에 가장 적합한 것이 무엇인지 확인하려면 네트워크 구성/계층 수를 테스트하는 것이 좋습니다.

이르판 2019년 3월 2일 오전 4시 40분 #

회신하다

안녕 제이슨,

놀라운 작업입니다. 노이즈 제거와 관련된 신호 처리 부분에 대한 링크가 있나요? 3차 중간 필터와 3차 저역 통과 버터워스 필터(차단 주파수 = 20Hz)와 같은 필터는 어떻게 되나요? 또한 여기서 정규화가 어떻게 이루어지는지도 알려주세요.

Jason Brownlee 2019년 3월 2일 오전 9시 36분 #

회신하다

죄송하지만, 저는 그 주제에 대한 튜토리얼이 없습니다. 어쩌면 DSP 라이브러리를 살펴보는 게 어떨까요? 심지어 scipy도 이런 걸 어느 정도 구현할 수 있습니다.

Manisha 2019년 3월 26일 오전 11시 11분 #

회신하다

안녕하세요 선생님,
데이터 세트를 조금 설명해 주세요. 제가 이해하기 어려워서요...
제가 이해한 바를 말씀드리겠습니다 ::::
acc와 gyro 센서에서 6개의 시계열이 있습니다.
각 시계열을 128개의 샘플이 있는 창으로 나눕니다... 총 7352개의 포인트가 있으며 각 포인트는 하나의 활동에 해당합니다.
데이터 세트 설명에는 "각 활동 창을 설명하기 위해 총 561개의 피처가 추출되었습니다"라고 언급되어 있습니다. 여기서 헷갈립니다... 각 창에는 128개의 샘플이 있고 각 창은 561개의 피처를 나타냅니다... 여기서 헷갈립니다..

Jason Brownlee 2019년 3월 26일 오후 2시 17분 #

회신하다

예.

추출된 특징은 데이터 세트의 일부로 사용할 수 있지만, 반드시 사용할 필요는 없습니다. 사실, 저는 대부분의 예에서 특징을 사용하지 않습니다.

Manisha 2019년 4월 4일 오후 2시 41분 #

회신하다

안녕하세요 선생님,

"10명의 피험자에 대한 전체 가속도 데이터의 히스토그램"에서 Y축은 무엇을 나타냅니까?
2500 5000 등?... X축은 데이터가 조정되므로 -1에서 1까지입니다. Y축은 어떻습니까?

Jason Brownlee 2019년 4월 5일 오전 6시 10분 #

회신하다

좋은 질문이네요. 히스토그램은 x축에 대한 관찰 빈도를 세는 것이라는 걸 기억하세요.

Khan 2019년 10월 23일 오후 5시 49분 #

회신하다

안녕하세요 선생님,
저는 두 개의 센서 "가속도계"와 "자이로스코프"에서 데이터를 수집했습니다. 이 데이터에서 단일 모델을 훈련하고 싶습니다. "UCI" 또는 제 작업을 충족하는 다른 접근 방식과 같이 데이터 세트를 구축하는 방법을 안내해 주시겠습니까? 감사합니다.

Jason Brownlee 2019년 10월 24일 오전 5시 36분 #

회신하다

UCI 데이터세트의 예를 시작점으로 삼고 귀하의 필요에 맞게 조정할 수 있을까요?

Khan 2019년 10월 24일 오후 5시 8분 #

회신하다

하지만 가속도계와 "자이로스코프"는 다른 값을 제공합니다. 가속도계는 1초에 12개의 값을 제공하는 반면, 자이로스코프는 24개의 값을 제공합니다. 그래서 UCI와 같은 데이터 세트를 매핑하고 구성할 수 없습니다. 이에 대해 안내해 주세요. 감사합니다.

Khan 2019년 10월 24일 오후 5시 11분 #

회신하다

UCI 센서 데이터세트와 같은 데이터세트를 구성하는 데 도움이 되는 리소스를 찾지 못했습니다.

Jason Brownlee 2019년 10월 25일 오전 6시 36분 #

회신하다

그렇게 하려면 사용자 지정 코드를 작성해야 할 수도 있습니다. 천천히 진행하고 다른 데이터 세트를 준비하기 위해 코드를 활용해보세요.

너무 힘들다면, 필요한 일을 해줄 개발자를 고용하는 게 어떨까요? upwork.com에서 말이에요?

Jason Brownlee 2019년 10월 25일 오전 6시 35분 #

회신하다

어쩌면 데이터 크기를 조정해야 할 수도 있겠죠?

Khan 2019년 10월 31일 오전 1시 28분 #

회신하다

좋아요

Khan 2019년 10월 31일 오전 1시 40분 #

회신하다

저는 50Hz 데이터 세트에서 "가속도계"와 "자이로스코프"에 대한 Lstm 모델을 훈련했습니다. 이제 이 모델을 배포하여 예측을 얻고 싶습니다. 훈련을 위해 다음을 설정했습니다.

시간 단계 = 100
세그먼트 시간 크기 = 180

이제 실제로 테스트 데이터 세트를 전달하여 훈련된 모델에서 예측을 얻는 방법을 생각하고 있습니다. 예측을 얻고 잘못된 결과를 피하는 정확한 방법은 무엇일까요?

이 프로젝트에 사용 중인 저장소에 대한 링크를 따르세요

https://github.com/bartkowiaktomasz/har-wisdm-lstm-rnns

내 질문이 이해되지 않는다면... 내 질문을 반복할 수 있습니다.

Jason Brownlee 2019년 10월 31일 오전 5시 33분 #

회신하다

model.predict()를 호출하여 새로운 데이터에 대한 예측을 할 수 있습니다.

아마도 이것이 도움이 될 것입니다:
https://machinelearningmastery.com/how-to-make-classification-and-regression-predictions-for-deep-learning-models-in-keras/

Khan 2019년 10월 31일 오후 6시 13분 #

회신하다

알았어, 네 말은 알겠어

Jason Brownlee 2019년 11월 1일 오전 5시 26분 #

회신하다

감사해요!

Khan 2019년 10월 31일 오후 6시 18분 #

회신하다

하지만 LSTM에서처럼 정의된 세그먼트 크기로 데이터를 전달하면 예측 후 해당 세그먼트의 결과가 반환됩니다. 그래서 세그먼트 크기를 전달하는 정확한 방법을 생각하고 있는데, 그러면 정확한 결과를 얻을 수 있습니다.

Jason Brownlee 2019년 11월 1일 오전 5시 27분 #

회신하다

죄송합니다. 질문을 이해할 수 없습니다. 자세히 설명해 주시겠습니까?

Aggelos Papoutsis 2019년 11월 11일 오후 8시 42분 #

회신하다

안녕하세요, 원시 데이터에서 파이썬 코드를 통해 윈도우를 계산하는 예가 있나요? 또한 원시 데이터에서 계산된 수작업 피처의 예가 있나요?

Jason Brownlee 2019년 11월 12일 오전 6시 37분 #

회신하다

네, 여기서 시작해 보세요:
https://machinelearningmastery.com/start-here/#deep_learning_time_series

AGGELOS PAPOUTSIS 2019년 11월 12일 오후 11시 49분 #

회신하다

감사합니다. 다른 질문을 해서 제 직감을 확인할 수 있을까요? 데이터가 10초 동안 50Hz에서 수집되었다는 것을 알고 있다면, 윈도우의 지속 시간을 어떻게 정의할 수 있을까요? 저는 문헌 참조(1초에서 2초 사이)를 사용하고 있고, 2초를 선택한다고 합시다. 그러면 2초 윈도우에 100개의 샘플이 있을 것입니다.

Jason Brownlee 2019년 11월 13일 오전 5시 43분 #

회신하다

이 모델은 타이밍에 구애받지 않으며, 입력 샘플만을 알고 있습니다.

입력 데이터에 대해 다양한 창 크기를 테스트하고 결과를 비교해 보는 건 어떨까요?

AGGELOS PAPOUTSIS 2019년 11월 13일 오후 5시 22분 #

고맙습니다, 제이슨. 딥 러닝의 프라이버시 기술에 대해 알고 계신가요? 차등 프라이버시 또는 페더레이션 러닝?

Jason Brownlee 2019년 11월 14일 오전 7시 58분 #

아니요, 죄송합니다.

zizomar 2019년 11월 20일 오전 3시 40분 #

회신하다

안녕하세요 여러분, 저는 방금 머신 러닝을 배우기 시작했고 CSV 파일 데이터(내 스마트폰으로 기록한 가속도계의 Z축)에서 EMD 분해를 실행하는 데 도움이 필요합니다.

Jason Brownlee 2019년 11월 20일 오전 6시 21분 #

회신하다

EMD 분해란 무엇입니까?

Sonia Sanju 2019년 11월 28일 오전 9시 56분 #

회신하다

선생님, 저는 삼성 갤럭시 노트3에서 수집한 원시 가속도계 데이터로 실험하고 있습니다. 파이썬에서 적용할 수 있는 데이터를 사전 처리하는 가장 좋은 필터는 무엇입니까? 감사합니다.

Jason Brownlee 2019년 11월 28일 오후 1시 32분 #

회신하다

여러 가지 접근 방식을 시도해 보고 특정 데이터 세트에 가장 적합한 성능을 제공하는 방법을 선택하는 것이 좋습니다.

zaid 2019년 12월 12일 오후 10시 15분 #

회신하다

안녕하세요,
저는 스마트폰을 사용하여 데이터를 수집했습니다. 정상 및 비정상 움직임의 두 가지 유형에 따라 데이터에 레이블을 지정하고 싶습니다.
출력에 레이블을 지정하는 방법과 노이즈를 필터링하는 방법을 알려주시겠습니까?

감사합니다

Jason Brownlee 2019년 12월 13일 오전 6시 2분 #

회신하다

처음에는 수동으로 라벨을 지정한 다음, 어떻게 자동화할 수 있는지 살펴보세요.

Sharmistha Das 2020년 1월 2일 오후 3시 35분 #

회신하다

안녕하세요! 코드를 알려주세요. 인간 인식 정확도를 어떻게 알 수 있을까요?

Jason Brownlee 2020년 1월 3일 오전 7시 15분 #

회신하다

아마도 여기서 시작해 보세요:
https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/

Amir Hussein 2020년 1월 22일 오후 9시 25분 #

회신하다

@Jason Brownlee 박사님, 이 모든 놀라운 튜토리얼에 감사드립니다. 각 축에 대해 별도의 분포를 생성하여 각 활동에 대한 세 축(x,y,z)의 데이터를 비교하는 방법을 이해했지만, 두 개의 다른 장치에서 가속도계 데이터가 있고 이 두 장치의 데이터 분포를 비교하고 싶은 경우, 세 축(x,y,z)을 장치에서 수집한 데이터를 나타내는 하나의 분포로 결합할 수 있을까요? 모든 활동의 데이터와 세 축의 데이터를 장치에 해당하는 하나의 분포로 요약할 수 있을까요?

Jason Brownlee 2020년 1월 23일 오전 6시 31분 #

회신하다

두 세트의 관찰을 합치는 것은 의미가 없다고 생각합니다. 필요하다면 두 가지를 모두 모델에 입력할 수 있습니다.

Amir Hussein 2020년 1월 25일 오전 3시 49분 #

회신하다

제이슨 박사님, 답변 감사합니다. 혼란을 드려 죄송합니다. 제 문제는 모델에 데이터를 공급하는 것이 아니라 두 가지 다른 기기(스마트폰 또는 스마트워치)의 가속도계 데이터 분포를 비교하는 것입니다. 기본적으로 저는 한 기기에서 개발한 다음 다른 새 기기에 배포하면 모델 성능이 저하되는 이유를 시각적으로 설명하려고 합니다. 이 기사를 바탕으로 한 제 아이디어는 세 축 데이터의 크기를 평균화한 다음 각 기기에 대한 히스토그램 또는 상자 그림을 계산하는 것입니다. 이것이 합리적이라고 생각하십니까? 무엇을 제안하시겠습니까?

Jason Brownlee 2020년 1월 25일 오전 8시 42분 #

회신하다

두 표본 분포를 비교하는 것은 좋은 시작처럼 들립니다. 예를 들어 통계적 가설 검정을 실시하는 것이 좋습니다.

Amir Hussein 2020년 1월 28일 오후 9시 39분 #

네, 제 경우에는 통계적 테스트가 더 의미가 있을 것 같아요. 팁을 주셔서 감사합니다 😊

Brittany Smith 2020년 2월 18일 오후 12시 55분 #

회신하다

그러면 주어진 데이터 세트로 HAR을 어떻게 예측할 수 있을까?

제이슨 브라운리 2020년 2월 18일 오후 1시 46분 #

회신하다

다음은 예입니다:
https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/

Yifan Du 2020년 5월 29일 오후 7시 #

회신하다

안녕하세요, 프로. 제이슨, 저는 항상 당신의 튜토리얼에서 많은 도움을 받고 있습니다. 요즘은 인간 활동 인식에 대한 프로젝트를 하고 있는데, 시계열 데이터를 필터링하기 위해 몇 가지 필터를 사용하고, fft, PSD 등과 같은 몇 가지 변환을 했습니다. 그러는 동안, 저는 데이터를 윈도우로 슬라이스했지만 정규화하지 않았고, 그런 다음 sklearn에 MLP를 가져왔습니다. 하지만, 저는 그것이 특징 추출 데이터보다 시계열에서 더 나은 성과를 냈다는 것을 발견했습니다. 정말 저를 많이 혼란스럽게 했는데, 제가 특징 추출하는 방법이 잘못되었다는 뜻인가요? 시간 내주셔서 감사합니다. 답변해 주시면 정말 감사하겠습니다.

Jason Brownlee 2020년 5월 30일 오전 5시 56분 #

회신하다

아마도 이는 귀하의 데이터 세트와 선택한 모델의 속성일 것입니다.

데이터 세트에 대해 더 자세히 알아보려면 대체 모델과 데이터 준비를 탐색해 보세요.

DS 초보자 2020년 6월 24일 오후 10시 36분 # 회신하다

브라운리 박사님, "6. 한 피험자에 대한 시계열 데이터 플롯"에 대해 설명해 주시겠습니까? 값 341은 무엇을 나타내는지(그리고 시계열 축 x가 350으로 가는 이유) – 그리고 피험자에 따라 달라집니다. 제가 알기로 이 축은 항상 128개의 데이터 포인트가 됩니다.

감사합니다!

제이슨 브라운리 2020년 6월 25일 오전 6시 20분 # 회신하다

저는 우리가 모든 데이터 또는 대부분의 데이터를 중복되지 않은 상태로 표시하고 있다고 생각합니다.

sonal 2020년 8월 5일 오전 2시 54분 # 회신하다

안녕하세요, 저는 이렇게
큰 판다스 데이터 프레임 데이터를 가지고 있습니다 : data('x', 'y', 'z', '레이블 데이터') 저는 이 데이터 프레임에 50% 겹치는 슬라이딩 윈도우를 적용하고 싶습니다. 그리고 각 윈도우에서 공통적인 특징(평균, 표준 편차, 분산)을 추출하고 싶습니다.

몇 가지 예를 들어 설명해 주세요.

제이슨 브라운리 2020년 8월 5일 오전 6시 18분 # 회신하다

여기를 참조하세요:
https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

Elbohy, Abdallah 2020년 12월 5일 오후 2시 33분 # 회신하다

안녕하세요, 제이슨 교수님.
X_train과 관성 신호의 차이점은 무엇인지, 그리고 왜 관성 신호는 로드하는 반면 X_train에서는 그렇게 하지 않으셨는지 궁금합니다.

X_train은 3번째 ButterWorth 필터로 분리하기 전의 데이터인 것 같나요?

제이슨 브라운리 2020년 12월 6일 오전 6시 57분 # 회신하다

게시물에서: ”

사전 처리된 데이터가 포함된 "관성 신호" 폴더.

Elbohy, Abdallah 2020년 12월 7일 오후 9시 16분 # 회신하다

X_train이란 무슨 뜻인가요? 무엇이 들어 있나요? 쓰여진 파일 'READ ME.txt'에는 561개의 엔지니어링된 특징이 들어 있습니다. 그러니까 X_train은 "raw_data"를 전처리하기 전이라는 뜻이 아닌가요?

제이슨 브라운리 2020년 12월 8일 오전 7시 42분 # 회신하다

훈련 데이터 세트.

Elbohy, Abdallah 2020년 12월 8일 오후 9시 7분 # 회신하다

선생님, 사실 저는 이해하지 못했습니다. 저는 훈련 세트가 무슨 뜻인지 알고 있습니다.
하지만 실제로 X_train과 inertial_Signals의 핵심 차이점은 무엇입니까?
inertial_Signals는 훈련 세트에서 전처리를 한 결과입니까?
X_train과 inertial_Signals의 관계는 무엇입니까?

제이슨 브라운리 2020년 12월 9일 오전 6시 17분 # 회신하다

내부 신호에는 사전 처리된(원시) 데이터가 포함되어 있으며, 이 튜토리얼에서 이를 사용합니다.

X_train.txt와 y_train.txt에는 이 튜토리얼에서 사용하지 않는 엔지니어링된 기능이 포함되어 있습니다.

차이점은 "3. 데이터 세트 다운로드" 섹션에 설명되어 있습니다.

Elbohy, Abdallah 2020년 12월 8일 오후 10시 #

회신하다

관성 신호에서 다른 계산을 수행하여 훈련 세트를 추출하는 것 같습니다.
맞나요?

제이슨 브라운리 2020년 12월 9일 오전 6시 19분 #

회신하다

내부 신호는 원시 데이터입니다.

계속해서 문제가 발생하면 원본 논문을 읽어보세요.

리아사트 2021년 3월 26일 오후 2시 40분 #

회신하다

훌륭하게 설명하셨습니다. 이 프로젝트가 마음에 드신다면 비디오 링크를 첨부해 주세요.

Jason Brownlee 2021년 3월 29일 오전 5시 50분 #

회신하다

감사해요.

영상이 없습니다.

Diana 2021년 9월 2일 오전 4시 23분 #

회신하다

안녕하세요 브라운리 박사님,

유익하고 체계적인 기사를 제공해주셔서 감사합니다. 저는 현재 단일 웨이트 리프팅 운동(5가지 양질의 수업)의 품질 성과를 실제로 예측해야 하는 프로젝트를 진행 중입니다.

저는 원시 센서 데이터를 사용하고 있으며, 움직임 데이터의 시간 단계를 고려하여 활동 품질을 예측하는 것으로 프로젝트를 구성할 계획입니다. 이것도 시계열로 처리해야 할까요?

나는 데이터를 무작위로 분할하여 검증 및 훈련을 시켰고 랜덤 포레스트 모델을 사용하여 ~97%의 높은 정확도를 얻었지만 이것이 내 결정이 옳다는 것을 의미하지는 않는다는 것을 이해합니다.

Jason Brownlee 2021년 9월 2일 오전 5시 15분 #

회신하다

네, 시계열처럼 들립니다.

관찰 결과가 시간적으로 정렬되어 있는 경우, 워크포워드 검증이나 유사한 방법을 사용하여 모델을 평가해야 합니다.
https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

Diana 2021년 9월 4일 오전 8시 17분 #

회신하다

감사합니다!

Jason Brownlee 2021년 9월 5일 오전 5시 9분 #

회신하다

천만에요!

DSNC 2023년 2월 11일 오후 8시 48분 #

회신하다

안녕하세요 제이슨,

먼저, 이 글을 공유해 주셔서 감사드리고 싶습니다.

그리고 활동별 센서의 히스토그램에 대해 묻고 싶습니다. 3가지 신호(total_acc, body_acc 및 body_gyro) 모두에서 y축과 z축 신호(녹색과 주황색)가 활동 1(걷기)에서 누락된 이유는 무엇입니까? x축 신호(파란색)만 활성화된 것으로 보입니다.

감사해요,

James Carmichael 2023년 2월 12일 오전 9시 31분 #

회신하다

안녕하세요...질문 감사합니다. 히스토그램을 검토해 보겠습니다. 코드를 실행했나요? 그렇다면 히스토그램이 다릅니까?

DSNC 2023년 2월 13일 오후 7시 31분 #

회신하다

안녕하세요 제임스,

코드를 실행하였고 동일한 결과를 얻었습니다.
하지만 적어도 제가 아는 바에 따르면 x와 y는 활동 #1의 히스토그램에 나타나야 할 값을 가지고 있습니다.

아직 해결책을 찾아볼 기회가 없었습니다.

만약 그렇다면, 혹은 현재 히스토그램이 정확하다고 생각하신다면, 알려 주시면 감사하겠습니다.

감사해요,

댓글을 남겨주세요

이름 (필수)

이메일 (공개되지 않음) (필수)

댓글 제출

환영합니다!
저는 *Jason Brownlee* 박사
이고 **개발자들이 머신 러닝** 으로 결과를 얻도록 돕습니다 .
더 읽기

튜토리얼을 놓치지 마세요:

당신을 위해 선택했습니다:

시계열 예측을 위한 LSTM 모델을 개발하는 방법

시계열 예측을 위한 합성 신경망 모델을 개발하는 방법


전력 사용을 위한 다단계 LSTM 시계열 예측 모델


인간 활동 인식을 위한 1D 합성 신경망 모델


Keras에서 LSTM을 사용한 다변량 시계열 예측

튜토리얼이 마음에 드시나요?

시계열을 위한 딥러닝 전자 책에서 ***정말 좋은*** 내용을 찾을 수 있습니다.

>> 내부 내용 보기

Machine Learning Mastery는 사람들이 기술을 알아내는 데 도움을 주는 데 중점을 둔 선도적인 디지털 미디어 퍼블리셔인 Guiding Tech Media의 일부입니다. 당사의 사명과 팀에 대해 자세히 알아보려면 회사 웹사이트를 방문하세요 .