pedometer.py

1) 프로그램 수행 (입/출력, 핵심 동작 파트)

입력은 파이썬의 sys 모듈을 사용하여 명령 프롬프트를 통해 받았습니다.정보파일에서 출력한 파일을 accel이라는 리스트에 한줄씩 분리해서 넣었습니다.

극댓값과 극솟값을 찾기 위해 각 가속도와 그 양옆의 값을 비교하여 앞의 가속도보다 작고 같고 뒤의 가속도보다 작으면 극솟값, 앞의 가속도보다 크고 뒤의 가속도보다 작거나 같으면 극댓값으로 분류하였습니다. 걸음수 판별의 위해 극솟값으로 시작하는 값은 큰 의미가 없기에 극솟값으로 시작하는 경우는 첫값을 제외하였습니다. 극솟값과 극댓값을 각각의 리스트로 만든 후 각 극댓값과 극솟값의 사이에 0을 지나는 값이 있다면 극값 리스트에 추가하였습니다.

걸음수를 판별하기 위해 각 상태를 지칭하는 함수를 만들어 극댓값과 극솟값을 인수로 받아 상태를 return했습니다. 함수를 이용하여 극값 리스트에 대해 함수를 반복시켜 해당 상태가 나오는 수를 측정하여 걸음수를 측정하였습니다.

상태구분자를 구분하기 위해 딕셔너리를 만들어서 key는 상태구분자와 value값에는 해당 상황의 가속도 값을 저장하였습니다. 해당 인덱스의 state 함수 리턴값이 run 이면 value값에 2를 저장, walk이면 1을 저장, stand 이면 0을 저장하였습니다.

출력은 output.txt 라는 파일명으로 위의 세줄은 걸음수와 아랫줄은 상태구분자와 상태를 표기하였습니다.

2) 걸음 수 세기 및 사용자 상태 (걷기/달리기/멈춤)를 분류하는 핵심 아이디어

```
#유효 극값 분리 파트
#극댓값 찾기
extreme= [] #극값 list
extreme_small = [] #극솟값 list
extreme_big = [] #극댓값 list
for i in range(1,len(accel)-1):
    if accel[i]<=accel[i-1] and accel[i]<accel[i+1]:
        extreme_small.append(i)
    if accel[i]>accel[i-1] and accel[i]>=accel[i+1]:
        extreme_big.append(i)

if extreme_small[0]<extreme_small[0]:
    del extreme_small[0] #극대,극소의 배열로 만들기 위해 첫 극값이 극솟값이라면 첫 극솟값은 제외

#극대, 극소의 순서로 두 값이 0값을 지날때 극값 list에 더하기
for i in range(min(len(extreme_small),len(extreme_small))-1):
    if accel[extreme_big[i]]*accel[extreme_small[i]]<0:
        extreme.append(extreme_big[i])
        extreme.append(extreme_small[i])

l = len(extreme)
```

우선 정보파일에서 극값을 분리합니다. 해당 순간의 가속도가 앞뒤 순간의 가속도보다 작으면 extreme_small, 크면 extreme_big 리스트에 추가합니다. 걸음수는 0에서 시작하여 극댓값을 찍고 극소값을 찍고 다시 0값이 될 때 하나의 걸음으로 측정하기 때문에 첫 극값이 극솟값인 경우에는 걸음수에 포함하지 않습니다. 그렇기에 극솟값과 극댓값을 통합한 extreme 리스트를 (극대, 극소)의 순서가 번갈아 나오게 하기 위해선 시작이 극솟값인 경우, 첫 극솟값은 제외합니다.

```
#걸음수 파트
#걷기상태 판별 함수
def state(b,s): #extreme 값중 accel index 값으로 받기
    if accel[s]<-6 or accel[b]>6:
        state = 'run'
    elif accel[s]<=-0.5 or accel[b]>=0.5:
        state = 'walk'
    else:
        state = 'stand'
    return state

#걸음수 측정
count_run=0 #달리기 횟수
count_walk=0 #걷기 횟수

for i in range(1,l//2 +1):
    a = state(extreme[2*i-2],extreme[2*i-1]) #홀수 인덱스는 극댓값, 짝수인덱스는 극솟값
    if a =='run':
        count_run+=1
    if a == 'walk':
        count_walk +=1

count_total =count_run+count_walk #총 걸음수
```

걷기 상태 판별을 위한 함수 state는 연속된 극댓값과 극솟값을 인수로 받아서 마지막 상태 state를 리턴합니다. 걷기 상태를 판별하고 하는 극댓값 (b)와 극솟값(s)의 구간에서 각 b와 s의 가속도 값의 절댓값 중 하나가 6보다 크다면 state를 'run'으로, 6과 0.5 사이라면 'walk'로, 둘다 아니라며 'stand'로 리턴합니다.

걸음수를 측정하기 위해 우선 걷기수는 count_walk, 달리기 수는 count_run 라는 변수를 지정합니다. 앞서 extreme 리스트의 길이를 l이라고 지정했습니다. extreme리스트 값은 홀수 인덱스는 극댓값을, 짝수인덱스는 극솟값을 나타냅니다. 그러기에 l을 2로 나눈 몫에 1을 더해주어 그 수만큼 for loop를 반복시켜줍니다. state 함수의 b위치에는 extreme의 홀수 인덱스를, s 위치에는 extreme의 짝수 인덱스를 넣어 리턴값이 'run'이면 count_run변수에 1을 더해주고, 리턴값이 'walk'이면 count_walk변수에 1을 더해줍니다. 총 걸음수를 의미하는 count_total 변수는 count_walk의 값과 count_run의 값을 더해준 값을 저장해줍니다.

3) 예제에 수행 결과

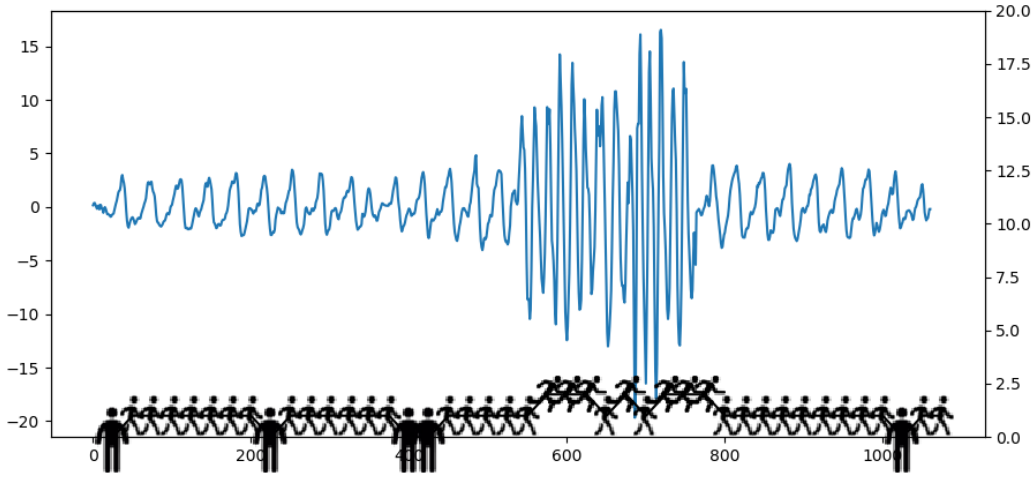
1. Mixed

output_mixed.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

38
24
14
S0: 1
S1: 1
S2: 1
S3: 0
S4: 1
S5: 1
S6: 0
S7: 1
S8: 1

총걸음수 : 38, 걷기 :24, 달리기: 14



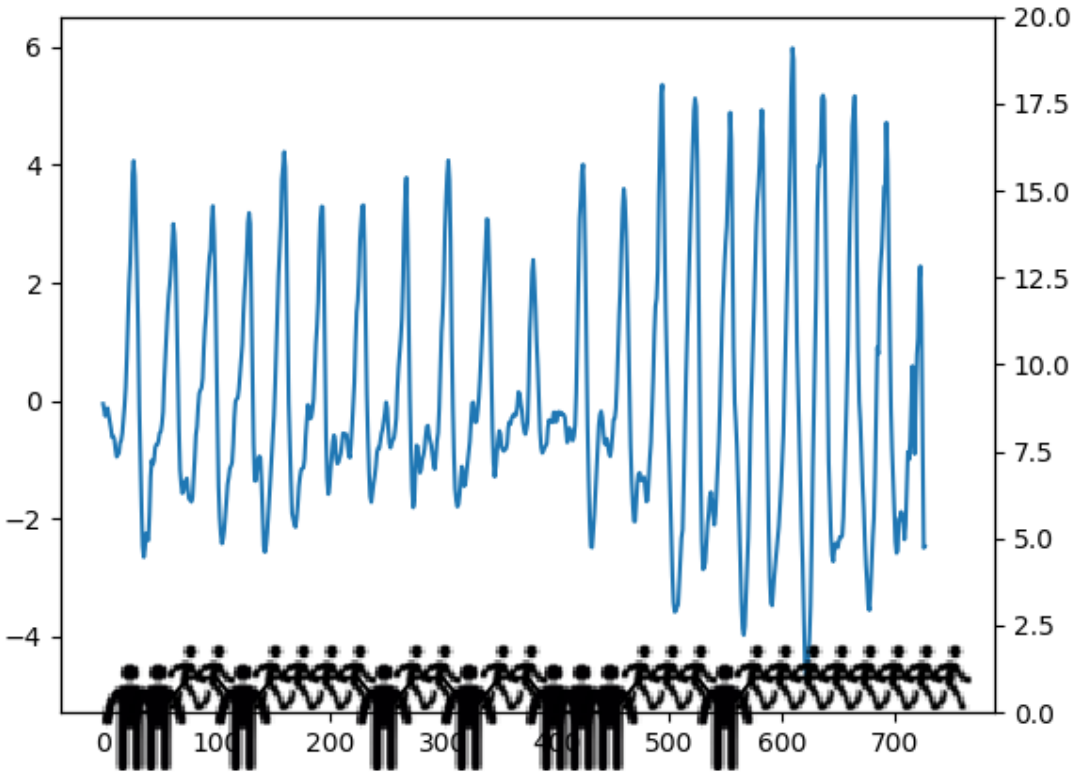
2. Sample1

output_sample1.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

23
23
0
S0: 0
S1: 0
S2: 1
S3: 1
S4: 0
S5: 1
S6: 1
S7: 1
S8: 1

총걸음수 : 23, 걷기 :23, 달리기: 0



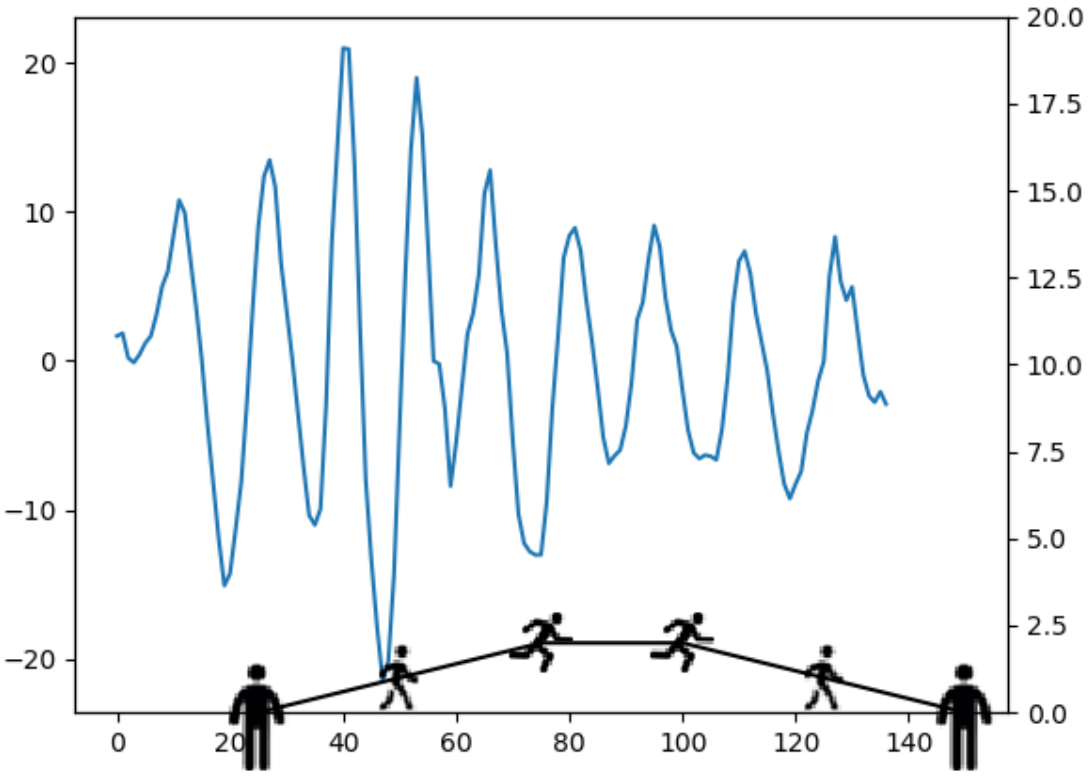
3. run_sample

output_run_sample.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H

9
1
8
S0: 0
S1: 1
S2: 2
S3: 2
S4: 1
S5: 0

총걸음수 : 9, 걷기 :1, 달리기: 8


20
10
0
−10
−20
0
20
40
60
80
100
120
140
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0