논문 읽기

형태학적

Morphological features

Using the QRS summits (after normalization), the maximum of cross-correlation function between each detected beat and the following beat was calculated, as well as the maximum of cross-correlation between the current beat and the previous beat detected, called respectively Corr1 and Corr2 [17]. Another feature was the maximum of cross-correlation between a template of normal beat, with each QRS complex detected, called CxyCxy, was computed. For each record, the template was calculated as the averaged beat of a sequence of many normal sinus beats.

Corr1
현재와 그다음
beat간 cross-correlation
의

Corr2
현재와 이전
beat간 cross-correlation
의 maximum

Cxy Cxy
normal과 현재의
cross-correlation의
maximum

Cross Correlation

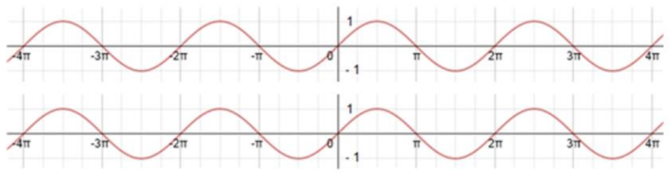
: 신호간 닮았는지 비교할수 있는 방법

$f \star g[n] = \sum_{m=-\infty}^{\infty} f^*[m] g[m+n]$

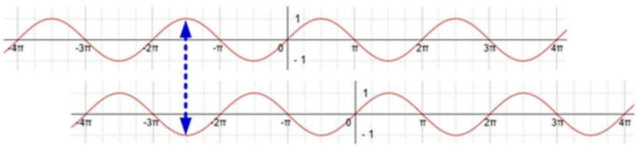
(어떤 신호는 고정 시켜 놓고 다른 신호는 시간 축으로 한개씩 움직여서 몽땅 두신호를 곱해서 더해보기)
f^* g

→ 신호가 비슷하다면 큰 값!

ex)



위 그림과 같이 동일한 sine 함수이면 CorrCoef(상관계수) = 1



위 그림과 같이 정 반대인 경우 CorrCoef = −1

numpy 로 cross correlation 구하기 기초

[깃허브 코드 보기]

→ One Note 정리

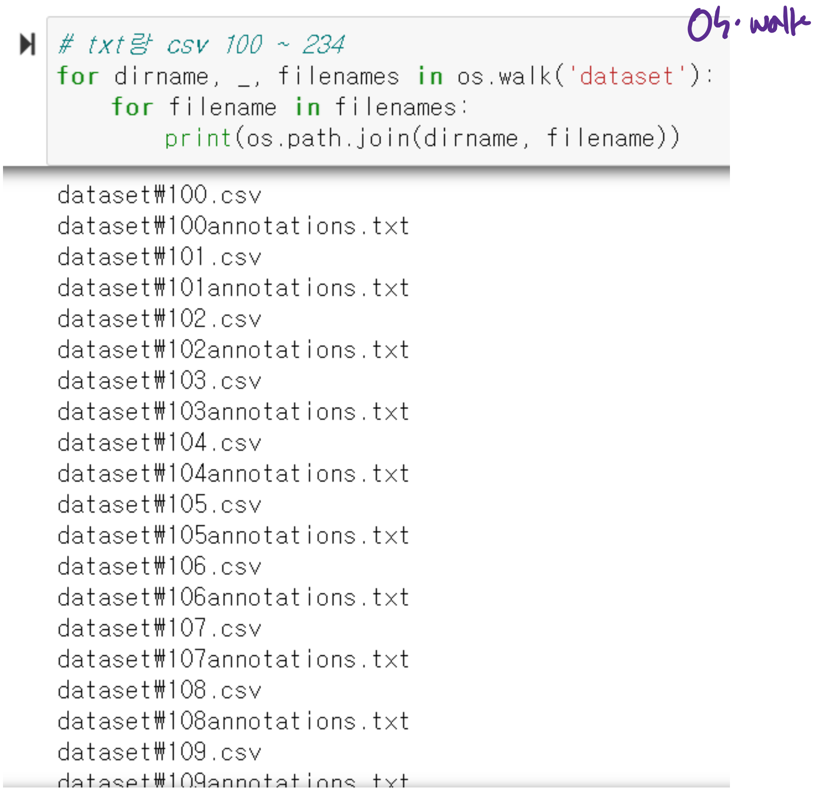
[Cross Correlation] ([웹 보기])

신호 값들 배열들

np.corrcoef(x, y, z)

$$\rightarrow \begin{pmatrix} \text{x와 x 상관계수} & \text{x와 y ~} & \text{x와 z ~} \\ \text{y와 x ~} & \text{y와 y ~} & \text{y와 z ~} \\ \text{z와 x ~} & \text{z와 y ~} & \text{z와 z ~} \end{pmatrix}$$

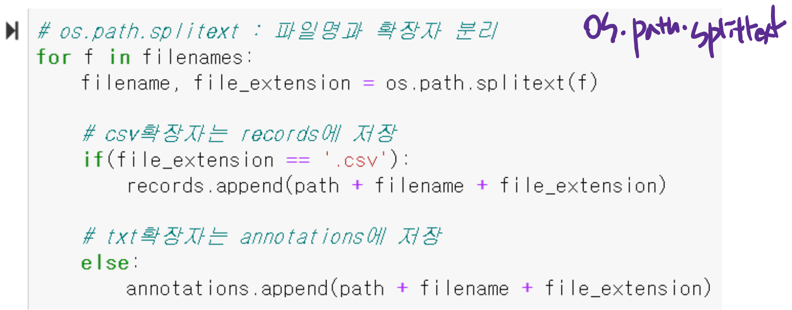
csv 파일들 불러와서 정리하기



Os. walk

```
# txt랑 csv 100 ~ 234
for dirname, _, filenames in os.walk('dataset'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
dataset₩100.csv
dataset₩100annotations.txt
dataset₩101.csv
dataset₩101annotations.txt
dataset₩102.csv
dataset₩102annotations.txt
dataset₩103.csv
dataset₩103annotations.txt
dataset₩104.csv
dataset₩104annotations.txt
dataset₩105.csv
dataset₩105annotations.txt
dataset₩106.csv
dataset₩106annotations.txt
dataset₩107.csv
dataset₩107annotations.txt
dataset₩108.csv
dataset₩108annotations.txt
dataset₩109.csv
dataset₩109annotations.txt
```

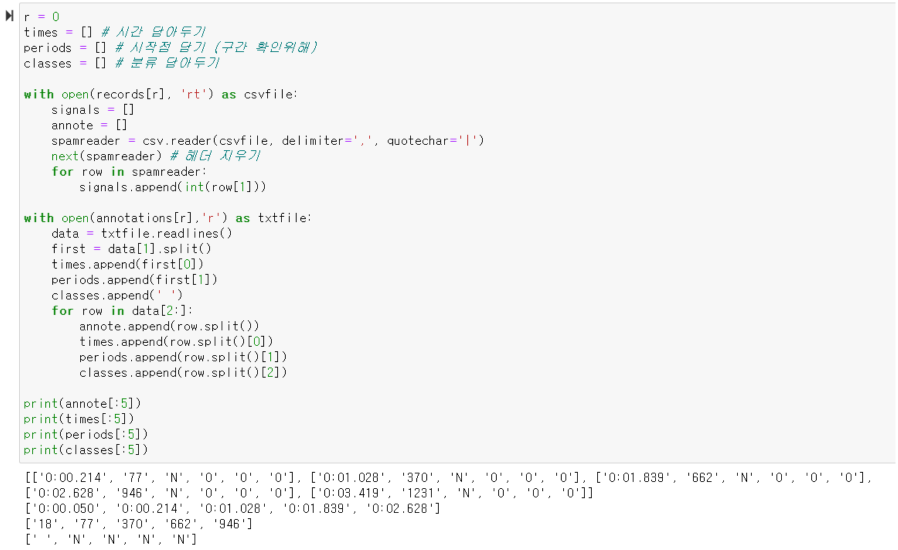


Os. path. splitext

```
# os.path.splitext : 파일명과 확장자 분리
for f in filenames:
    filename, file_extension = os.path.splitext(f)

    # csv확장자는 records에 저장
    if(file_extension == '.csv'):
        records.append(path + filename + file_extension)

    # txt확장자는 annotations에 저장
    else:
        annotations.append(path + filename + file_extension)
```



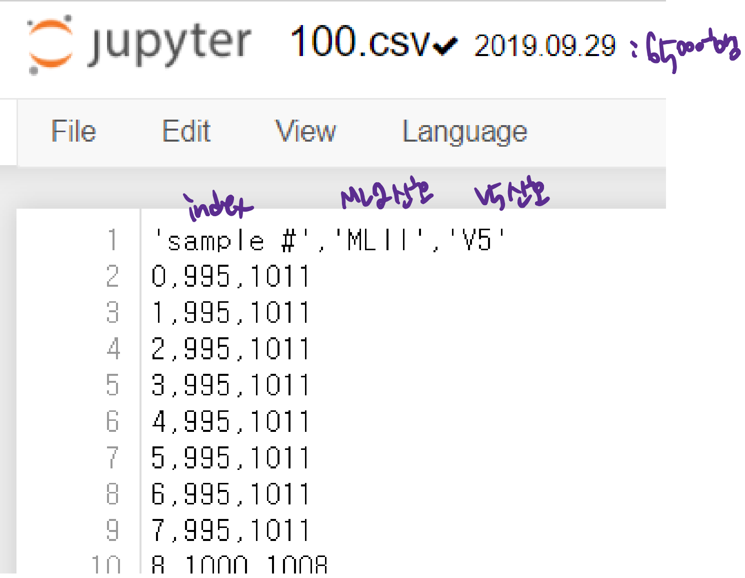
```
r = 0
times = [] # 시간 담아두기
periods = [] # 시작점 담기 (구간 확인위해)
classes = [] # 분류 담아두기

with open(records[r], 'rt') as csvfile:
    signals = []
    annote = []
    spamreader = csv.reader(csvfile, delimiter=',', quotechar='|')
    next(spamreader) # 헤더 지우기
    for row in spamreader:
        signals.append(int(row[1]))

with open(annotations[r],'r') as txtfile:
    data = txtfile.readlines()
    first = data[1].split()
    times.append(first[0])
    periods.append(first[1])
    classes.append(' ')
    for row in data[2:]:
        annote.append(row.split())
        times.append(row.split()[0])
        periods.append(row.split()[1])
        classes.append(row.split()[2])

print(annote[:5])
print(times[:5])
print(periods[:5])
print(classes[:5])
```

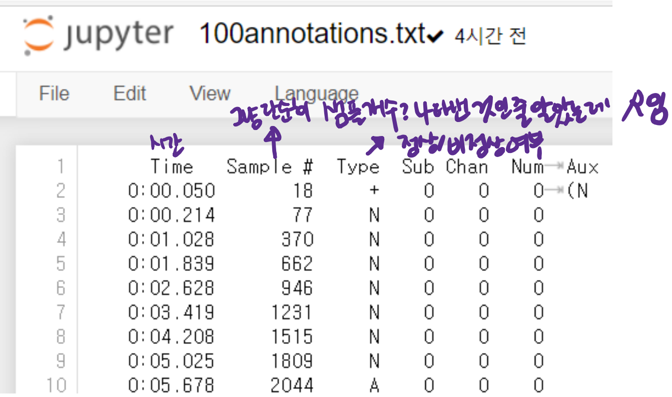
```
[['0:00.214', '77', 'N', '0', '0', '0'], ['0:01.028', '370', 'N', '0', '0', '0'], ['0:01.839', '662', 'N', '0', '0', '0'],
['0:02.628', '946', 'N', '0', '0', '0'], ['0:03.419', '1231', 'N', '0', '0', '0']]
['0:00.050', '0:00.214', '0:01.028', '0:01.839', '0:02.628']
['18', '77', '370', '662', '946']
[' ', 'N', 'N', 'N', 'N']
```



jupyter 100.csv✔ 2019.09.29 : 65000행

index MLⅡ신호 V5신호

```
'sample #','MLII','V5'
0,995,1011
1,995,1011
2,995,1011
3,995,1011
4,995,1011
5,995,1011
6,995,1011
7,995,1011
8,1000,1008
```



jupyter 100annotations.txt✔ 4시간 전

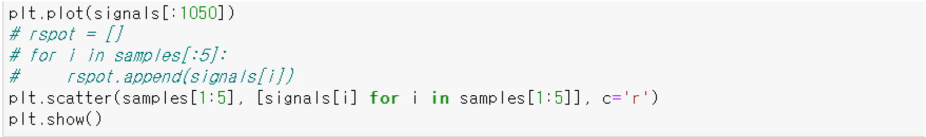
그냥 단순히 샘플개수? 나해본 것인줄 알았는데 R용

시간

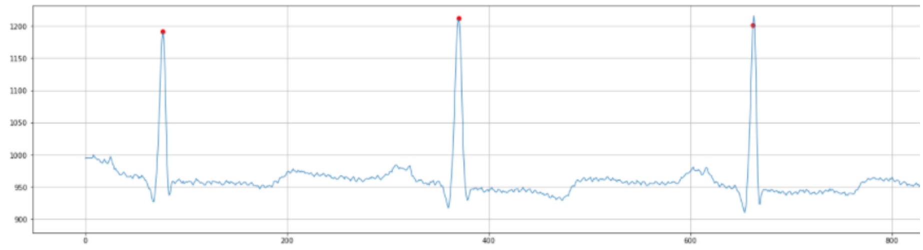
정상/비정상여부

```
      Time   Sample #  Type  Sub Chan  Num→Aux
  0:00.050       18     +     0    0    0→(N
  0:00.214       77     N     0    0    0
  0:01.028      370     N     0    0    0
  0:01.839      662     N     0    0    0
  0:02.628      946     N     0    0    0
  0:03.419     1231     N     0    0    0
  0:04.208     1515     N     0    0    0
  0:05.025     1809     N     0    0    0
  0:05.678     2044     A     0    0    0
```

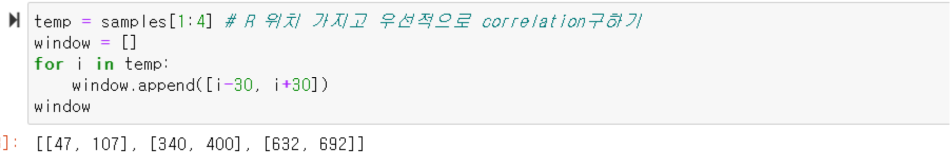
3 개의 beat 로 corrcoef 구하기



```
plt.plot(signals[:1050])
# rspot = []
# for i in samples[:5]:
#     rspot.append(signals[i])
plt.scatter(samples[1:5], [signals[i] for i in samples[1:5]], c='r')
plt.show()
```



samples에 담긴 R 위치 가지고 신호를 R위치로 잡아서 앞뒤 30개로 비교함 (즉, window 크기 60개)



```
temp = samples[1:4] # R 위치 가지고 우선적으로 correlation구하기
window = []
for i in temp:
    window.append([i-30, i+30])
window
```

```
]:  [[47, 107], [340, 400], [632, 692]]
```

```
In [100]:  for i in range(len(window)-1):
               array = np.corrcoef(signals[window[i][0]:window[i][1]], signals[window[i+1][0]:window[i+1][1]])
               print(array)

           [[1.         0.97521667]
            [0.97521667 1.        ]]
           [[1.         0.96604411]
            [0.96604411 1.        ]]
```

```
In [104]:  for i in range(len(window)-1):
               array = np.corrcoef(signals[window[i][0]:window[i][1]], signals[window[i+1][0]:window[i+1][1]])
               print(i+1,"번째 신호의 Corr1 : ",array[0][1])

           0번째 신호의 Corr1 :  0.97521666910301
           1번째 신호의 Corr1 :  0.9660441147874228
```

```
In [107]:  for i in range(len(window)-1):
               array = np.corrcoef(signals[window[i][0]:window[i][1]], signals[window[i+1][0]:window[i+1][1]])
               print(i+2,"번째 Corr2 : ",array[0][1])

           2 번째 Corr2 :  0.97521666910301
           3 번째 Corr2 :  0.9660441147874228
```

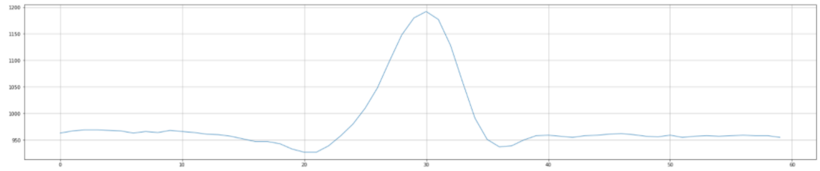
Corr1은 현재 beat과
다음 beat의 correlation,
Corr2은 현재 beat와
이전 beat의 correlation
이므로 당연히
1번째 신호 corr1
= 2번째 신호 Corr2 임

```
In [108]:  for idx, item in enumerate(classes[1:], start = 1): #8,A
               if item != 'N':
                   print(idx, item)
                   break;

           8 A
```

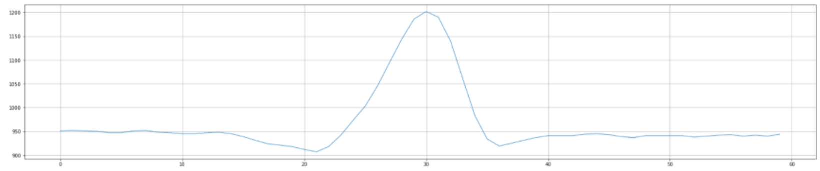
8번째가 'A'

```
# 그냥 첫번째 N신호 가져옴
normal_signal = signals[samples[1]-30:samples[1]+30]
plt.plot(normal_signal)
plt.show()
```



그냥 처음에 N이였어서 가져온 신호

```
abnormal_signal = signals[samples[10]-30:samples[10]+30]
plt.plot(abnormal_signal)
plt.show()
```



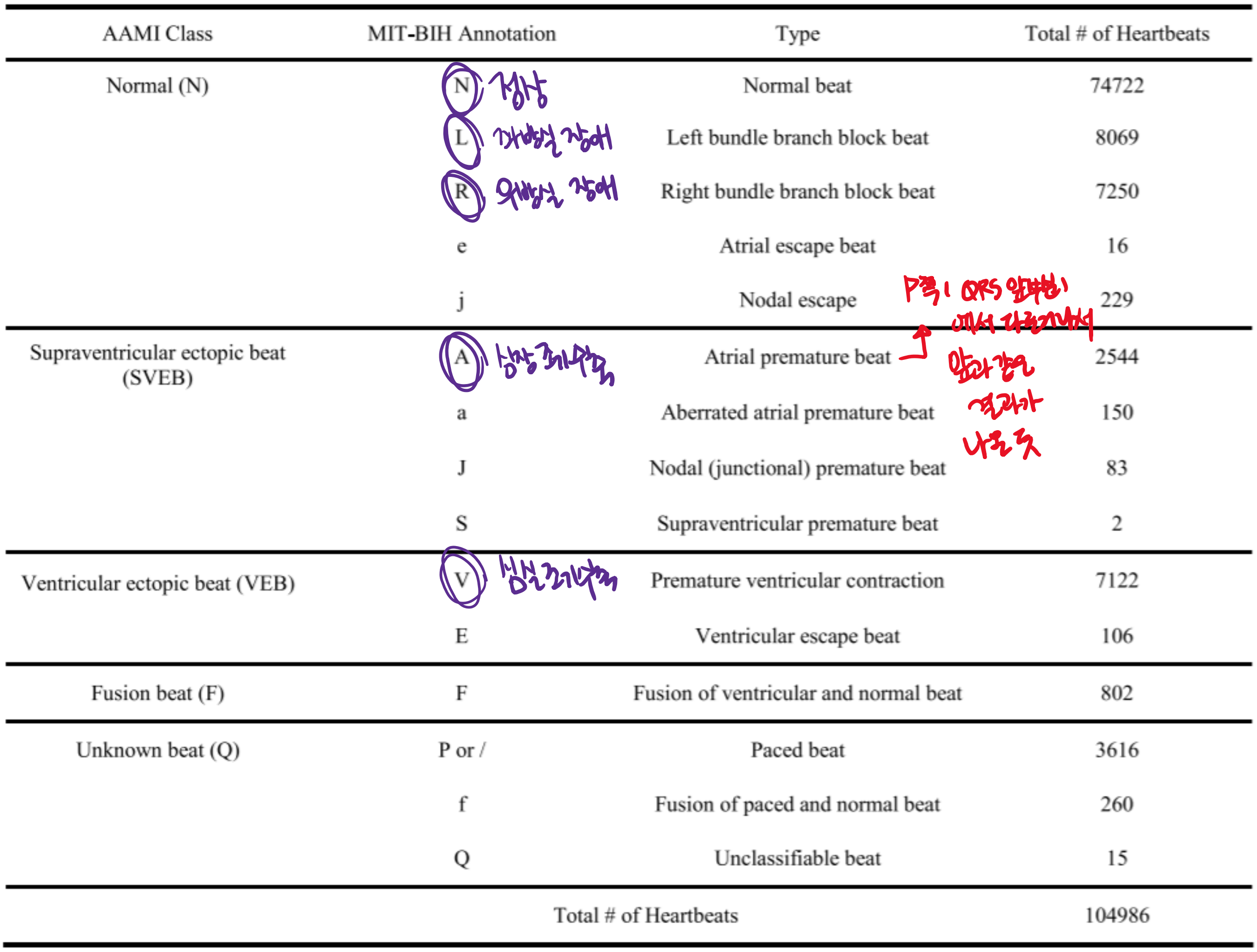
A인 신호

```
arr = np.corrcoef(normal_signal, signals[samples[8]-30:samples[8]+30])
print(arr[0][1])

0.9848637652319113
```

???? 근데 N끼리 비교보다
상관계수 값이 커서 이렇게 간단하게 생각하면 안되는 거 같음 ...

beat 별 분류클래스



AAMI Class	MIT-BIH Annotation	Type	Total # of Heartbeats
Normal (N)	N 정상	Normal beat	74722
	L 좌방실 장애	Left bundle branch block beat	8069
	R 우방실 장애	Right bundle branch block beat	7250
	e	Atrial escape beat	16
	j	Nodal escape	229
Supraventricular ectopic beat (SVEB)	A 심방조기수축	Atrial premature beat	2544
	a	Aberrated atrial premature beat	150
	J	Nodal (junctional) premature beat	83
	S	Supraventricular premature beat	2
Ventricular ectopic beat (VEB)	V 심실조기수축	Premature ventricular contraction	7122
	E	Ventricular escape beat	106
Fusion beat (F)	F	Fusion of ventricular and normal beat	802
Unknown beat (Q)	P or /	Paced beat	3616
	f	Fusion of paced and normal beat	260
	Q	Unclassifiable beat	15
		Total # of Heartbeats	104986

P파 I QRS 앞부분에서 다르게 나서 앞과 같은 결과가 나올 듯