Vad 算法加速实例

马振坤　　2020 年 4 月 8 日星期三

注：通过工具 line_profiler 找到函数中耗时最多的步骤，进行去 for 循环，实现算法加速。

Step1

针对 smooth 函数优化

```
# Before
def smooth_filter(data):
    '''
    filter audio data by smooth
    Parameters
    ----------
    data: numpy array of float32
        audio PCM data

    Returns
    ----------
    smooth_data: numpy array of float32
        audio PCM data
    '''
    count = len(data)
    smooth_data = np.array([0.0001 for i in range(count)])
    # print("count = ", count)
    for i in range(count-1):
        # smooth_data[i] = (data[i] + data[i+1] + data[i+2]) / 3
        smooth_data[i] = (data[i] + data[i + 1]) / 2
    return smooth_data
```

原作者显然对numpy不熟悉，直接对用array错位相加，再平均，用数组操作代替for循环，即可实现一样的效果。

对执行效率的影响：★★★★★

修改难度：无

# after
```
def smooth_filter(data):
    '''

    filter audio data by smooth
    Parameters
    ----------
    data: numpy array of float32
        audio PCM data

    Returns
    ----------
    smooth_data: numpy array of float32
        audio PCM data
    '''
    return np.append((data[:-1] + data[1:]) / 2, data[-1])
```

Step2

针对 audioFrameRMS 函数优化

# Before
```
# Compute audio root mean square in a frame
def audioFrameRMS(segment):
    sum_tmp = 0
    for i in range(len(segment)):
        sum_tmp += segment[i]**2
    sum_tmp = math.sqrt(sum_tmp/len(segment))
    return sum_tmp
```

直接对array平方，再开方，用数组操作代替for循环，即可实现一样的效果。

对执行效率的影响：★★★★★

修改难度：无

```
# after
def audioFrameRMS_m(segment):
    segment = segment*segment
    return np.sqrt(np.sum(segment)/len(segment))
```

Step3

针对 smooth 函数优化

```
# Before
def audioFrameZeroCrossingRate(segment, threshold=0.01):
    zeroCrossing = 0

    # segment = np.clip(segment, 0, 0.02)
    for i in range(1, len(segment)):
        # zeroCrossing += abs(sgn(segment[i]) - sgn(segment[i-1]))
        if (segment[i] >= threshold) & (segment[i-1] <=-threshold):
            zeroCrossing += 1
        elif (segment[i] <= -threshold) & (segment[i-1] >=
threshold):
            zeroCrossing += 1
    zeroCrossingRate = zeroCrossing / len(segment) / 2
    return zeroCrossingRate
```

先对array减去阈值，通过*threshold*来判断正负号，再对array错位相乘，判断变号的地方，用数组操作代替for循环，即可实现相似的效果。

对执行效率的影响：★★★★★

修改难度：★（需要知道布尔索引）

# after
```
def audioFrameZeroCrossingRate_m(segment, threshold=0.01):
    x = segment
    x[x >  threshold] = 1
    x[x < - threshold] = -1
    x = x.astype(np.int)
    y = x[:-1] * x[1:]
    z = y[y == -1]
    return - np.sum(z)/ len(segment)
```

Step4

针对数组优化

# Before
```
runningFrameFeature1 = np.array([])
runningFrameFeature1 = np.append(runningFrameFeature1, frameRMS)
```

python中，list的操作，比numpy中的array添加更简洁，速度更快，10倍的速度差。

对执行效率的影响：★★★
修改难度：★

# after
```
runningFrameFeature1 = []
runningFrameFeature1.append(frameRMS)
```