Feature engineering for breathing rate detection

Continuous detection of breathing or respiratory monitoring is important for many brunches of medicine, including detection of sleep apnea, for respiratory rehabilitation and detection of abnormal breathing patterns. Wearable technology is able to provide continuous and convenient way to measure noninvasive breathing rate measurements and monitoring.

It becomes important to develop precise algorithms and machine learning techniques that work on data produced by wearable technology. It processes respiratory raw signals and interprets them into useful information.

In this paper we will discuss the following components of the breathing rate detections:

- feature engineering to process signals from 3D axis accelerometer that are used by breathing detection algorithms.
- results from using extracted features on actual data
- one of the proposed algorithms for breathing detection

Note that the breathing detection algorithm we will discuss here compensates for some minor data loss.

First let's discuss specification of a physical sensor used to collect the data.

The Sensor Spec

The sensor is attached around abdominal error or tucked into waistband in order to provide a snug attachment to belly.

The signal used in this paper is produced by a Monbaby product by https://monbaby.com/. It is particularly designed for detection of rollovers and interruption in breathing movements in infants.

In this paper we will extend its usage and apply it for detection of breathing rate in older kids and adults.

The accelerometer sensor used in MMA8451Q 3-Axis, 14-bit. You can find a datasheet here,https://www.nxp.com/docs/en/data-sheet/MMA8451Q.pdf.

The sampling frequency of the 3D accelerometer signal is 6.25Hz or every 160ms.

Feature engineering

In order to produce useful information, one needs to filter out the noise from the signal, process the cleaned signal and extract from it the features that would would indicate presence of breathing, and vice versa absence of it.

The Signal Processing

The module cleans and processes the accelerometer data in blocks. Each data block is defined within a running window of about 20 seconds or around 125 data points.

The step immediately proceeding the data processing conducts data cleansing. The data is cleaned via application of data signal processing filters, specifically Butterworth band-pass filter. It filters out frequencies beyond 0.2-2Hz frequency band.

Next step is principe component analysis (PCA) applied on already filtered data to transform 3 input axis into into 3 components. Only first two most dominant components are used for further processing.

Fast Fourier transform (FFT) is applied on those two separated components. The breathing rate is calculated as most dominant by amplitude peak, by using find_peaks algorithm.

This is how we get our first two feature: first two breathing rates.

In order to detect signal quality and detect breathing interruption, we calculate goodness of fit and signal stability. Those become our next two features.

The goodness of fit is calculated in Chi squared fit value with one degree of freedom. The good signal is considered to be above 0.9 . The goodness of fit is calculated as the maximum between first two components of PCA. This is done to compensate a corner case when the signal separation by PCA did not work properly.

The stability of the signal is calculated as the 1st derivative of the filtered signal and difference in energy expenditure. If any data point in the sample of 20 seconds exceeds three times exponential weighted mean value (multiple of 3), then the sample is considered unstable.

```
x2 = df[xa]; x1 = df[xa].shift()
y2 = df[ya]; y1 = df[ya].shift()
z2 = df[za]; z1 = df[za].shift()
eed = np.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)+(z2-z1)*(z2-z2))
if self.meanDiff:
    jumps = eed>self.MAXDIFF*self.meanDiff
    numDiffs = jumps.sum()
if self.goodness>0:
    if self.meanDiff: self.meanDiff = 0.1*eed.mean() + 0.9*self.meanDiff
    else: self.meanDiff = eed.mean()
```

Alternatively, if any data point in 20 second sample exceeds 6 times of standard deviation of displacement:

```
xd = df['raw_x']-df['raw_x'].mean()
yd = df['raw_y']-df['raw_y'].mean()
zd = df['raw_z']-df['raw_z'].mean()
ed = np.sqrt(xd*xd+yd*yd+zd*zd)
if self.meanDisp:
    jumps = ed>self.MAXDISP*self.meanDisp
    numDisp = jumps.sum()
if self.goodness>0:
```

```
        if self.meanDisp: self.meanDisp = 0.1*ed.std() + 0.9*self.meanDisp
        else: self.meanDisp = ed.std()
```

The model description

The No-Breathing condition is detected when the goodness of fit is less than 0.9 AND the sample is stable. The pseudo-code logic is shown below:

```
         if np.any(instability>1):
            self.state = 0
            self._alarm = None
        else:
            if np.any(goodness<1.0):
                self.state = -1
                self._alarm = True
            else:
                self.state = 1
                self._alarm = False
```

Here is a more precise code sample:

```
          if self.iframe==0 or minInstability>self.STBTHR:
            self.state = 0
            self._alarm = None
        else:
            if ((minGoodnessScaled<self.GOODTHR and (len(duplicates)==0 or
not self.DUPTHR or maxDuplicates>self.DUPTHR2)) or \
                (len(duplicates)>0 and self.DUPTHR and
maxDuplicates>self.DUPTHR)) and \
                (not self.prevAmps or self.prevAmps/currAmp > self.MAXJUMP):
                self.state = -1
                self._alarm = True
                if minGoodnessScaled<self.GOODTHR: # and (len(duplicates)==0
or not self.DUPTHR or maxDuplicates>self.DUPTHR2):
                  log.warn('alarm @ipos:{}, goodness:{}<{}, dups:
{}'.format(self.ipos,minGoodnessScaled,self.GOODTHR,maxDuplicates))
                if self.DUPTHR and maxDuplicates>self.DUPTHR:
                  log.warn('alarm @ipos:{}, dups:{}
>{}'.format(self.ipos,maxDuplicates,self.DUPTHR))
            else:
                self.state = 1
                self._alarm = False
```

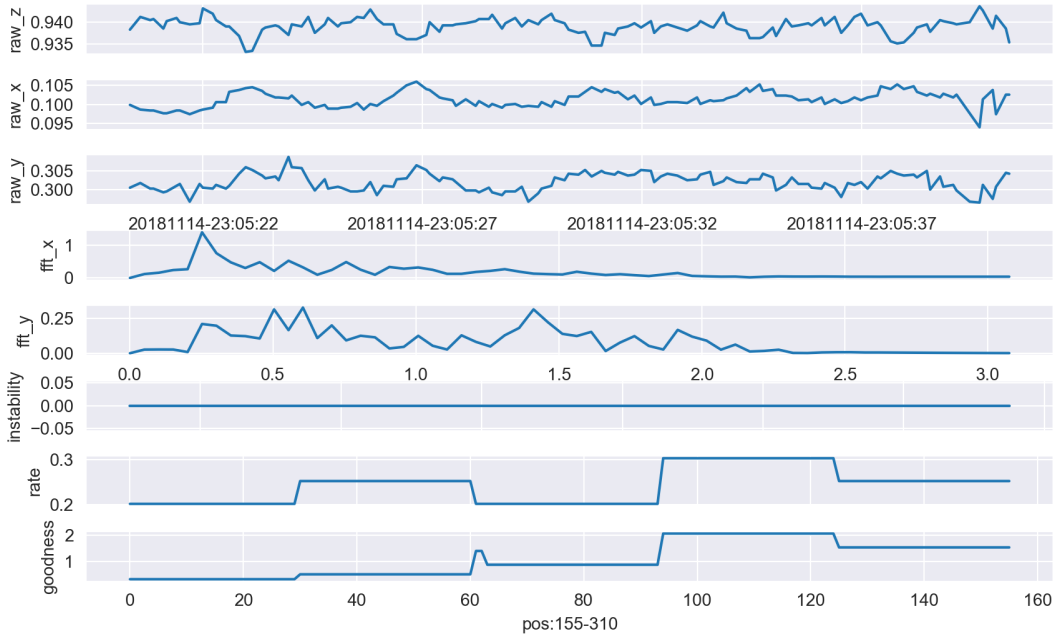
The model code calculates additional metrics 'entropy' and signal-to-noise ratio, that are not used in the algo. Entropy process is very similar to chi2 but does not have clear threshold, such as chi2.

The results

Here are resulting features extracted from 3D accelerometer signal:

1. 1st component frequency
2. 2nd component frequency
3. goodness of fit as chi-2
4. stability of the signal

A graph below depicts the actual data from accelerometer, the features described above and the actual measured rate.



The subplots in the graph above are as follows:

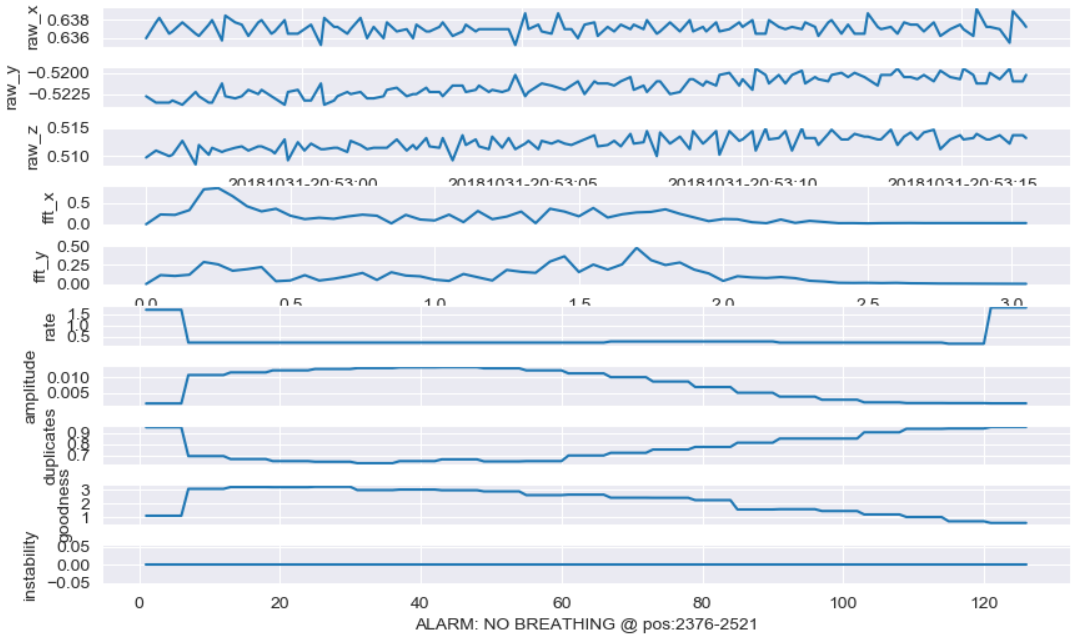
- raw_x - raw accelerometer signal from 1 axis
- raw_y - raw accelerometer signal from 1 axis
- raw_z - raw accelerometer signal from 1 axis
- fft_x - fft frequency and amplitude from 1st component
- fft_y - fft frequency and amplitude from 1st component
- instability - reverse of stability, if it's 0 then the signal is stable
- goodness - goodness of fit
- rate - measure breathing rate in Hz

The signal is actually collected from the actual adult subject and shows one rolling window of 20 seconds of breathing.

Below are results for various use cases including no breathing and baby breathing.

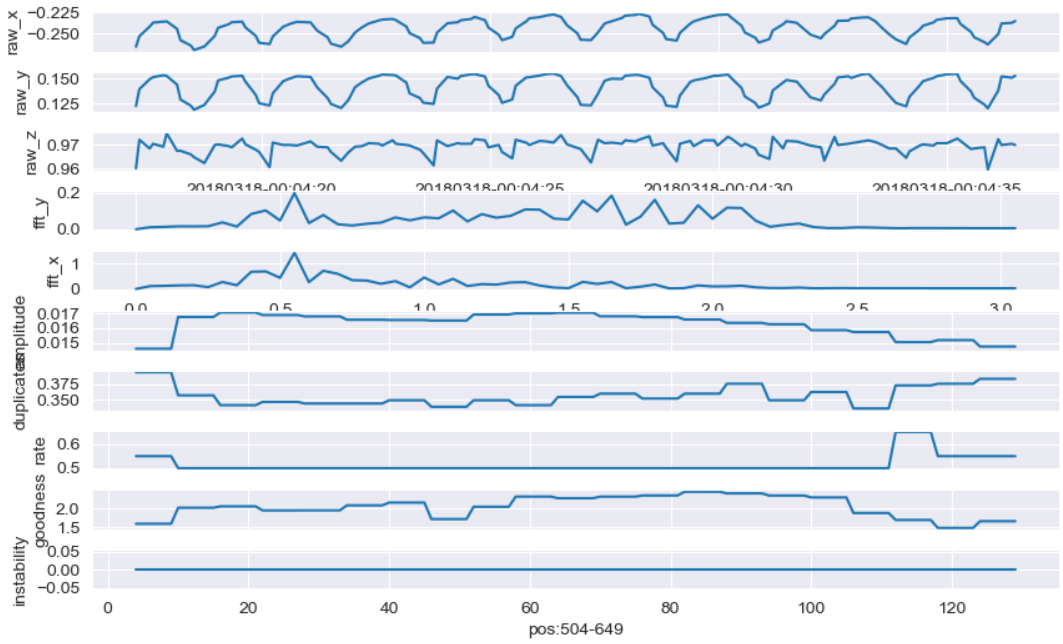
No Breathing

This is result of processing signal from the sensor held when an adult subject is holding the breath. The algorithm detects no breathing and shows it as "ALARM: NO BREATHING" depicted in the title.



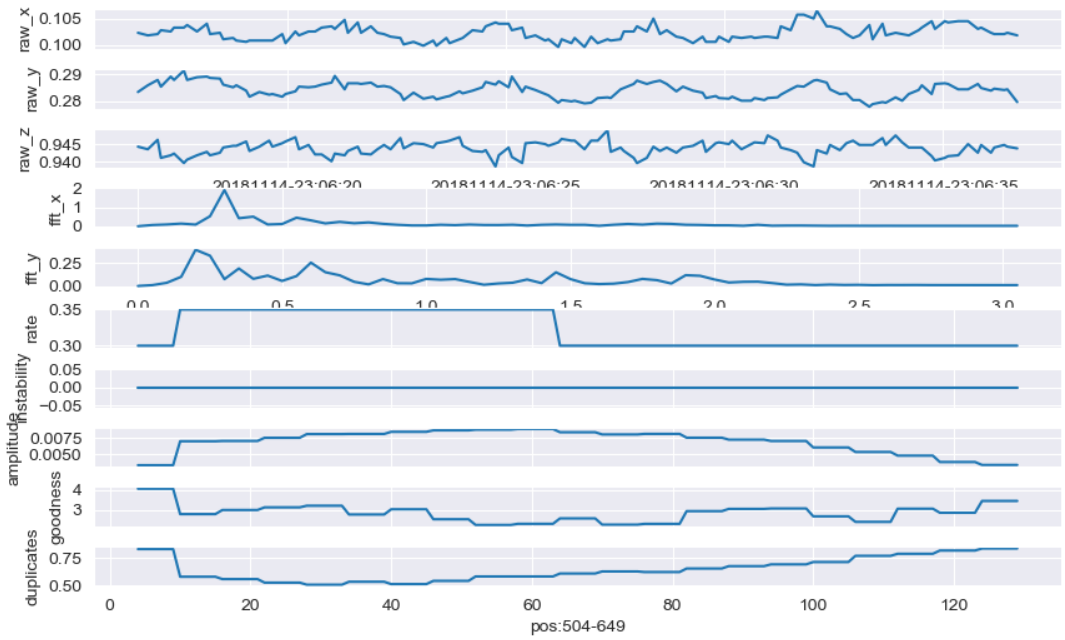
Baby Breathing

Actual signal from 12 month infant breathing. There were no baseline but the breathing rate varies between 0.5 and 0.6Hz (30-36 breath per minute), which is consistent with baby breathing rate of that age group and sufficiently more frequent that adult.



Adult Breathing

The signal from adult subject shows rate of 0.3-0.35Hz (18-21 breaths per minute) with 0.35Hz baseline.



The Proposal for Breathing Detection Rate

The main purpose of the algorithm is to classify the input signal into breathing, no breathing and unstable states. The actual breathing rate detection only considers stable regions. The region is stable if:

- there are no jumps and
- no large displacements from the mean.
- The signal classification is based on 3 features:
- actual breathing rate value
- ratio of duplicates to total number of values in the frame
- goodness of fit, which is chi2 fit of distribution to poisson, scaled by duplicate ratio above

the breathing detection algorithm we will discuss here compensates for some minor data loss by using the scaling of criteria based on the sample size in the running window:
The formula for scaling is

```
goodnessScaled = goodness / minimum of ratio of duplicates
within the 20 sec frame.
```

The alarm indicating interruption of breathing is triggered if either one of the following two condition is true:

- scaled goodness is less than a threshold and duplicate ratio is greater than 80%
- duplicate ratio is greater than 92% (see DUPTHR value above) AND

- there is a jump in magnitude between previous 20 second window and current window, defined as ration between max previous magnitude and min current magnitude that exceeds 8.5 times.

Conclusion

Note that the algorithm described above is one of many that were being tested and have not made the final cut.

Eventually we have selected another algorithm based on machine learning techniques but using the same set of features described above. The results of the applying one of the algorithms that used the features engineered and described above has created 98% breathing rate detection accuracy measured on a sample of 25 people when detecting accuracy of the breathing rate detection within 10% of the baseline. The baseline was measured with the NEULOG respiration belt monitor. We also measure the accuracy of breathing pauses detection and it came out 95% accurate in the same sample of 25.

It has been implemented as a feature in the commercial application: https://play.google.com/store/apps/details?id=com.monbaby001 and is currently being used by thousands of customers.