ВЫЯВЛЕНИЕ ГОЛОСОВОЙ АКТИВНОСТИ (VAD) С ИСПОЛЬЗОВАНИЕМ МЕЛ-КЕПСТРАЛЬНЫХ КОЭФФИЦИЕНТОВ

Чугунов Владимир

Санкт-Петербургский Государственный Университет (СПБГУ)

e-mail: inflamepheonix@gmail.com

Введение

Распознавание речи или Voice Activity Detection (VAD) – основных задача распознавания речи, применяема далее в кодировании речи (для последующей отправки), выделении диктора (из нескольких голосов) и, собственно, распознавании. В первом случае, VAD позволяет снизить объем передаваемых данных, исключая промежутки сигнала без речи из очереди отправки. Для двух других VAD работает на «чистотой» обрабатываемых моделей, а значит и всей системы в целом. Так же VAD используется в подавлении шума в VoIP-телефонии и в слуховых аппаратах.

Грубо говоря, её суть заключается в том, чтобы выделить речь одного диктора в всевозможных ситуациях: от пустой комнаты без постороннего шума до улицы, завода или торгового центра с посторонними голосами.

Всего существует три возможных пути решения: задание порогового значения, статистическое моделирование и машинное обучение. Первый способ хорош при низком фоновом шуме – SNR (Signal-to-noise ratio), но при высоком начинает допускать ошибки. Два других позволяют работать и при высоком SNR, и при низком, однако производительность в «плохом» случае резко падает, что неприемлемо, для, например, мобильных приложений.

Решение задачи

Для начала, я опишу, чем и почему могут быть полезны мел-кепстральные коэффициенты.

Сам звук является последовательностью колебаний воздуха, воспринимаемых, в дальнейшем, слуховым трактом человека. Однако давно известно, что наше ухо отнюдь не в одинаковой мере воспринимает звуки разных частот. В связи с этим была создана шкала громкости, основанная на **фон**ах – логарифмических единицах громкости:

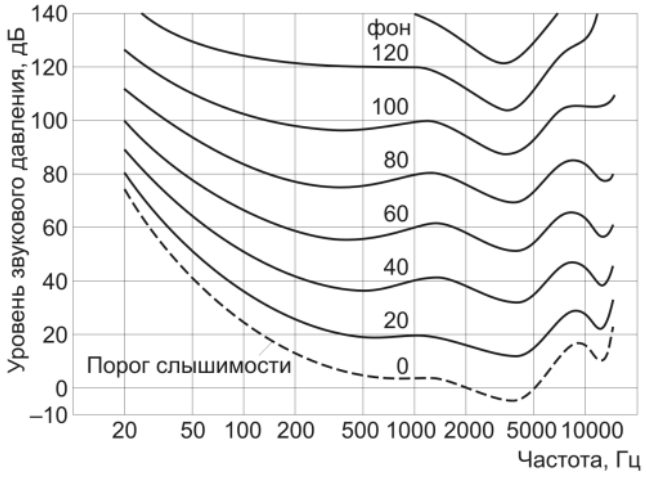

Рис.1 *Зависимость громкости от звукового давления.*

Так же, и воспринимаемая ухом информация зависит от частотных характеристик далеко не очевидным образом:

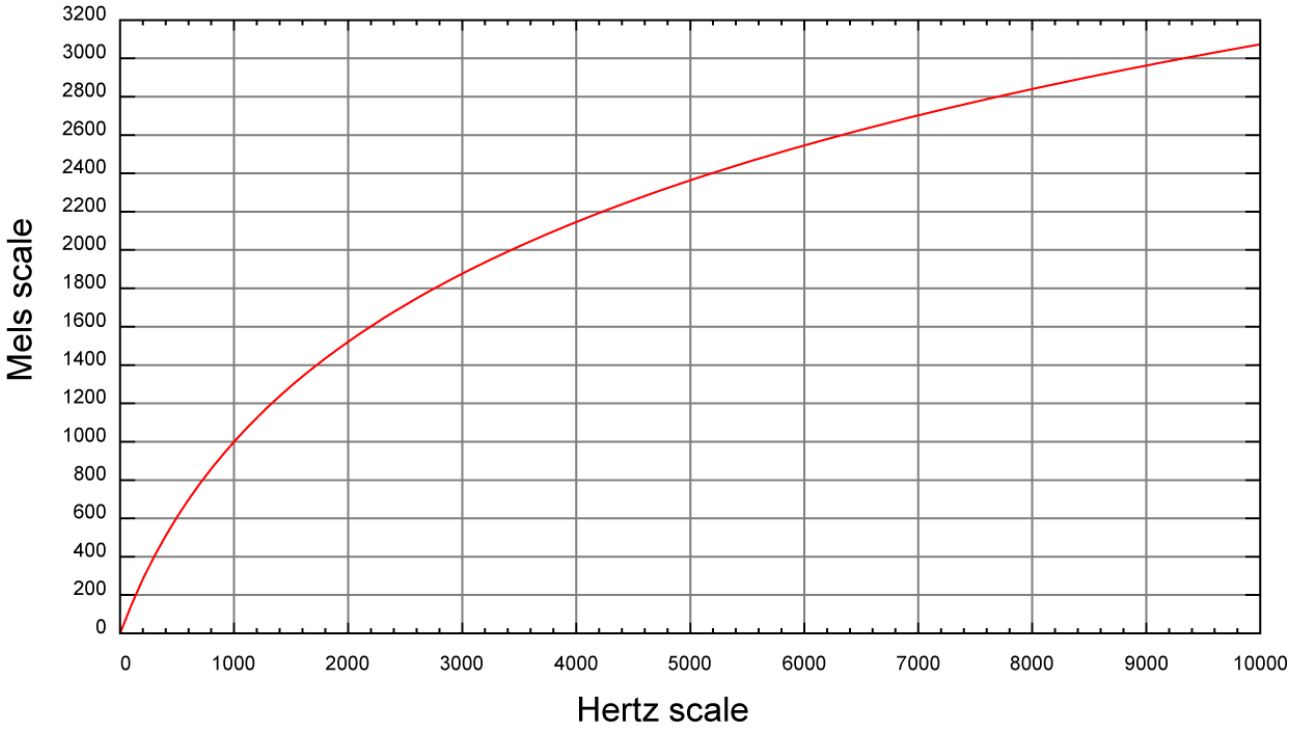

Рис.2 *Зависимость высоты звука в мелах от частоты колебаний*

Однако теперь можно легко вывести зависимость мелов от частоты в виде формулы (полученной экспериментально):

$$mel = 1127 \cdot \ln\left(1 + \frac{freq}{700}\right) = 2595 \cdot \log_{10}\left(1 + \frac{freq}{700}\right)$$

И обратная формула:

$$freq = 700 \cdot \left(e^{\left(\frac{mel}{1127}\right)} - 1\right) = 700 \cdot \left(10^{\left(\frac{mel}{2595}\right)} - 1\right)$$

Кепстр можно определить, как «энергетический спектр от прологарифмированного энергетического спектра сигнала»:

$$C(s) = |\mathcal{F}\{\ln(|\mathcal{F}\{x(s)\}|^2\}|^2 = \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} \ln(|\mathcal{F}\{x(s)\}|^2) \cdot e^{iwq} \cdot ds$$

Кепстральное преобразование, помогающее в решении нашей проблемы, так же широко используется и в работе над другими задачами, связанными с обработкой звука. Так, с помощью этого преобразования, можно определить основную частоту сигнала в пиках полученного кепстра (Рис.3).

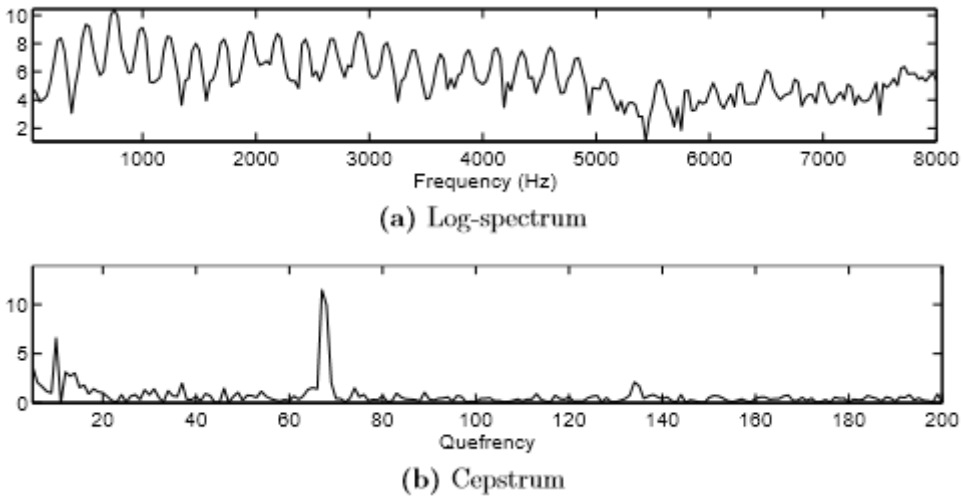

Рис.3 *Применение оценки кепстра*

Необходимая же нам область применения несколько уже. Мы будем использовать специальные банки мел-фильтров, примененные к определенным частотным окнам спектра.

Алгоритм

Для начала, нам нужно снять гармонический сигнал с микрофона, а точнее – его цифровое представление в виде массива байтов:

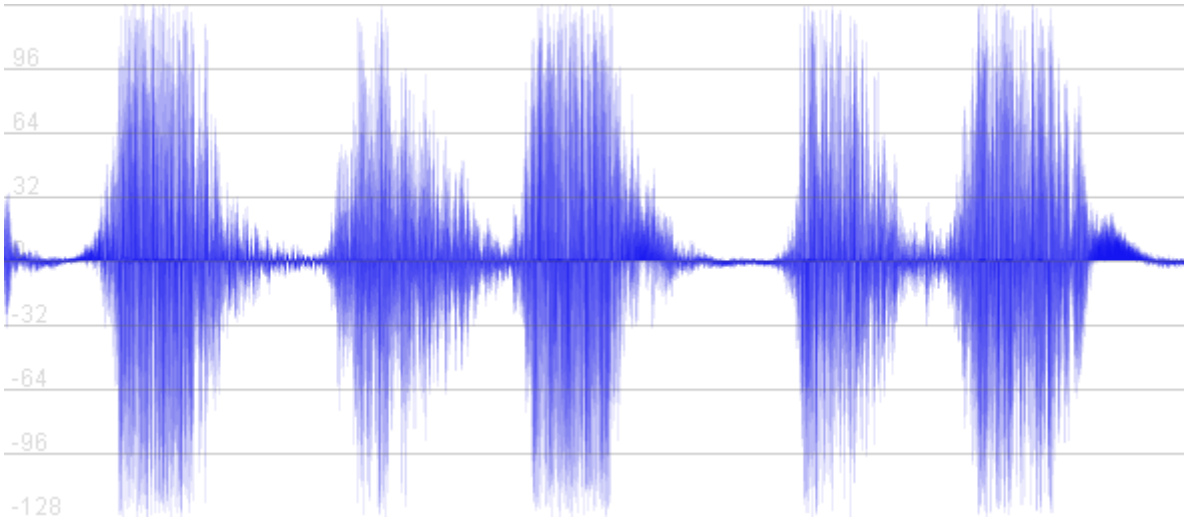

Рис.4 *Гармонический цифровой сигнал*

Первым делом нужно распределить сигнал по частотам. Здесь подходят несколько видов преобразование, основанных на дискретном преобразовании Фурье:

$$\mathcal{F}(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right)$$

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{\left(\frac{2\pi i}{N} \cdot kn\right)}, k = 0, 1, \ldots, N-1$$

Я работал только с реальной ($Re(z)$ для $z \in \mathbb{C}$) составляющей сигнала и использовал ортогональную энергию (normalized power) сигнала:

$$X_k = |\mathcal{F}\{x_k\}|^2 \; ; \quad X_k = \ln(|\mathcal{F}\{x_k\}|^2)$$

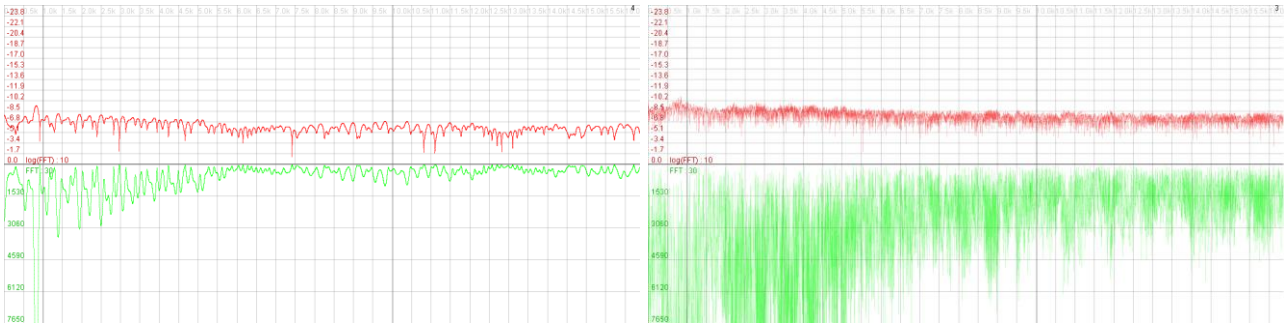

Рис.5 *Normalized power и логарифм от неё (оба - масштабированные) от сигнала. Слева – с применением оконной Hann-функции; справа – без.*
(Исходные сигналы правого и левого изображения различны)

Далее нам нужно создать банк мел-фильтров, то есть выделить определенные области в частотном диапазоне, с помощью которых мы будем определять MFCC (Mel-Frequency

Cepstral Coefficients). Определим границы фильтров. Для пояснения правильного распределения проще показать отдельный метод из реализации на языке Java:

```
private double[] getMelFilterBankBoundaries(double minFreq,
double maxFreq, int numberFilters){
    double[] centers = new double[numberFilters + 2];
    double maxFreqMel = linToMelFreq(maxFreq);
    double minFreqMel = linToMelFreq(minFreq);
    double deltaFreqMel = (maxFreqMel - minFreqMel)/(numberFilters + 1);
    double nextCenterMel = minFreqMel;

    for(int i = 0; i < centers.length; i++)
    {
      centers[i] = melToLinFreq(nextCenterMel);
      nextCenterMel += deltaFreqMel;
    }

    centers[0] =  minFreq;
    centers[numberFilters + 1] = maxFreq;

    return centers;
  }
```

Таким образом мы движемся по мел-частотной шкале линейно, а по частотной – логарифмически.

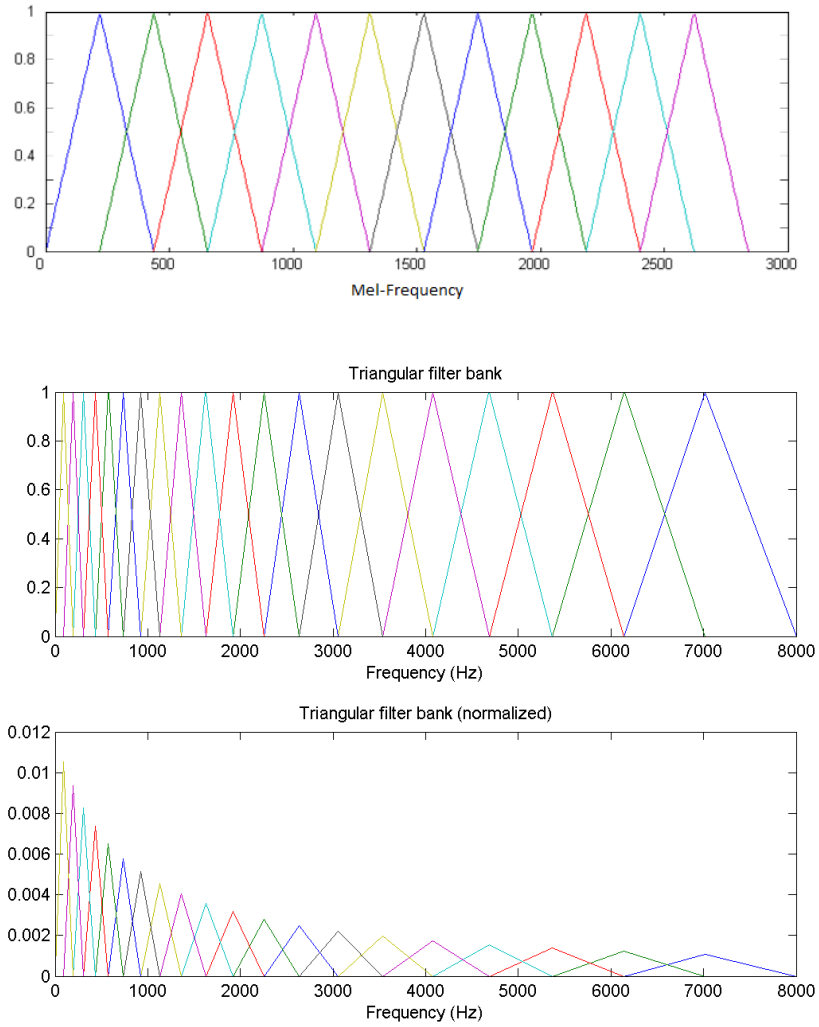

Рис. 6 *Отображение треугольных мел-фильтров на ось частот и мел-частот*

Далее для каждого окна мы рассчитываем фильтр:

```
for(int i = 1; i <= numberFilters; i++)
    {
      double[] filter = new double[(windowSize/2)+1];
      for(int j = 0; j < filter.length; j++)
      {
        double freq = baseFreq * j;
        filter[j] = getMelFilterWeight(i, freq, boundaries);
      }
      matrix[i-1] = filter;
    }
```

Где `getMelFilterWeight` возвращает весовое значение фильтра для `i`-го окна из их набора – `boundaries`, основанное на положении частоты `freq` относительно центра окна (получаем «восходящий» или «нисходящий» фильтр) и высоты фильтра, равной

$$height_i = \frac{2}{boundaries_{i+1} - boundaries_{i-1}}$$

Далее полученную матрицу банка мел-фильтров мы умножаем на вектор, ортогонализованной энергии обрабатываемого в данный момент окна (их размер и % перекрытия задаются перед началом просчета MFCC), логарифмируем полученный вектор и проводим дискретное косинусное преобразование (DCT; аналогично DFT без работы с мнимыми элементами) с помощью умножения на заранее подготовленную матрицу преобразования.

Таким образом, мы получаем все мел-кепстральные коэффициенты. Нулевой коэффициент играет роль энергии кратковременного промежутка:

$$E_t = \sum_t S(t) \otimes H(t);$$

где $S -$ сигнал, $H -$ спецефические характеристики фильтра, $s -$ время

Именно за ним нужно следить, если мы хотим попытаться узнать о появлении речи в сигнале. В дальнейшем анализе мы полагается на спектральную производную $\frac{d \ln S}{dt}$, опуская нулевой коэффициент. Дело в том, что нулевой коэффициент следит лишь за моментальным изменением спектра и никак не помогает отличать речь от других сигналов равных по силе.

Первый коэффициент отражает основные изменения спектра. Он особенно чувствителен к диктору и характеристикам шума. Следующие коэффициенты отражают тонкие детали спектра – энергетический баланс между различными полосами сигнала – формантами. В частности второй коэффициент отражает разницу между высокими и низкими частотами в спектре; третий является приближением суммы высоких и низких частот с вычитанием средних – того, что осталось между ними; четвертый – сумма высоких и низких и т.д. Коэффициенты 8-12 и выше уже сильно подвержены появлению различных артефактов, например при вычислении Hann-функции, так что имеют слишком малую точность в необходимом анализе и плохо применимы.

Заключение

В ходе работы был исследован и реализован метод распознавания наличия голосовой активности, обрабатывающий нулевой мел-кепстральный коэффициент. Как и ожидалось, он оказался вполне работоспособен.

Однако в связи с тем, на обработку алгоритму шел чистый сигнал с не самого лучшего микрофона, точность была не настолько велика, чтобы можно было использовать это в конечном продукте. Для полноценной работы, как VAD-модуля, нужно было бы добавить методы пропуска пауз по пороговой энергии (если энергия ниже заданной – считаем, что это тихий шум) для, например, исключения срабатывания модуля на громкое дыхание. Так же, текущий алгоритм не следит за длительностью активности, ведь вряд ли человек хотел тихо сказать что-то за 100ms, когда несколько секунд до и после этого молчал. Но он увидит возбуждение спектра, решит, что что-то важное произошло там и будет слушать какой-то шум. Например, на Рис.7 для алгоритма будут идентично «похожи» на речь фрагменты, выделенные желтыми прямоугольниками

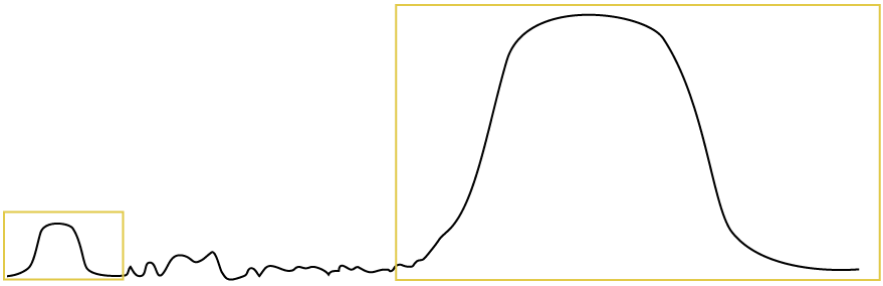

Рис.7 *Пример случая, в котором созданный алгоритм может ошибиться*

Использованная литература:

- Pham Chau Khoa (Ben), *"Noise Robust Voice Activity Detection"*, Nanyang Technological University, Singapore, 2012.
- Arnaud Martin, Delphine Charlet, Laurent Mauuary, *"Robust speech/non-speech detection using LDA applied to MFCC"*, France Te´le´com R&D, 2001
- Sergei Skorik and Frédéric Berthommier, *"On a cepstrum-based speech detector robust to white noise"*, Institut de la Communication Parlée (ICP/INPG), 10 Oct 2000
- Игорь *Урицкий*, *"Мел-кепстральные коэффициенты (MFCC) и распознавание речи"*, http://habrahabr.ru/post/140828/, 28 марта 2012