# Final Project

Matthew Markowitz, Lifu Xiao

November 13, 2018

## 1 Introduction

The database is a set of noisy recordings, which have poor quality for further usage. So it make sense to improve them. In order to remove the noise, we propose to use online dictionary learning.

## 2 Problem Statement

We used a python library known as librosa to import our audio. The audio recordings found in our test set had a sampling rate of 22050. This meant that every second of audio held approximately 22,000 numbers to represent it. For this reason, down sampling became necessary. Although some sacrifice in audio quality was necessary, we were able to reduce the sampling rate to 5,000, which made our calculations more feasible. The 5,000 points per second was still computationally intensive, but we found that we could break each second into X millisecond windows to ease computation further without sacrificing much quality. We found that a window size of 50 points or 50/5,000 = 1/100 second windows worked well for our dataset.

## 3 Algorithm

### 3.1 Data Preparation

Initializing the $\boldsymbol{A}_0 \in \mathbb{R}^{k\times k}$ and $\boldsymbol{B}_0 \in \mathbb{R}^{m\times k}$ as $\vec{\mathbf{0}}$
k is atoms number and m is the dictionary size.

### 3.2 Sparse Coding

When each $x_t$ come, using LARS to calculate

$$\alpha_t \triangleq \arg\min_{\alpha \in R^k} \frac{1}{2}\|\boldsymbol{x}_t - \boldsymbol{D}_{t-1}\alpha\|_2^2 + \lambda\|\alpha\|_1$$

where $\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{D} \in \mathbb{R}^{m \times k}$ and $t \leq T$(maximum number of iterations)
Then updating $\boldsymbol{A}, \boldsymbol{B}$ by

$$\boldsymbol{A}_t \leftarrow \boldsymbol{A}_{t-1} + \alpha_t \alpha_t^T$$

$$\boldsymbol{B}_t \leftarrow \boldsymbol{B}_{t-1} + \boldsymbol{x}_t \alpha_t^T$$

### 3.3 Dictionary Update

$$\boldsymbol{D}_t \triangleq \underset{\boldsymbol{D} \in C}{\arg\min} \frac{1}{t} \sum \frac{1}{2} \|\boldsymbol{x}_i - \boldsymbol{D}\alpha_i\|_2^2 + \lambda \|\alpha\|_1$$

Where $C \triangleq \boldsymbol{D} \in \mathbb{R}^{m \times k}$ s.t. $\forall j = 1, ..., k, \boldsymbol{d}_j^T \boldsymbol{d}_j \leq 1$ to ensure the convex.
Using block-coordinate descent to update dictionary Extracting columns of $\boldsymbol{A}$ and $\boldsymbol{B}$

$$\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_k] \in \mathbb{R}^{k \times k}$$

$$\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_k] \in \mathbb{R}^{k \times k}$$

for each column from $j = 1 \Rightarrow k$

$$\boldsymbol{u_j} \leftarrow \frac{1}{A[j, j]} (\boldsymbol{b}_j - \boldsymbol{D}\boldsymbol{a}_j) + \boldsymbol{d}_j$$

$$\boldsymbol{d}_j \leftarrow \frac{1}{\max(\|\boldsymbol{u}_j\|_2, 1)} \boldsymbol{u}_j$$

return $\boldsymbol{D}$ for next iteration

## 4 Experiments

### 4.1