Click Removal in Degraded Audio

Report for Module EEP55C22 Computational Methods

Changhong Li, Student ID 23333239
Trinity College Dublin
lic9@tcd.ie

October 2023

This report describes the algorithm designed for detection and removal of clicks in archived audio tracks. An example of a few clicks is shown in Figure 1.Clicks are like sudden, irrational spike in a waveform. The sound manifests as a short sharp click or thump in the audio track. These degradation arise because of the noise in the environment or something else. Our problem is to find out a model which could remove the clicks in the degraded signal and make the output signal more smooth.

1 Background

Let us define the observed degraded signal as G_k and the original, clean signal as y_k. We can model the degradation as follows.

$$G_k = \begin{cases} ccr_k & \text{when } b_k == 0 \\ y_k & \text{Otherwise} \end{cases} \tag{1}$$

where r_k is a random corruption generated with rand function as clicks array which have random locations and amplitude, and b_k is an indicator binary variable that is 1 at sites that are corrupted. Therefore the observed signal is created by corrupting specific sites indicated by $b_k == 1$, hence clicks shown in Figure 2.

Our problem is to train a model using the degraded signal which could predict how it should be, and then calculate the error between degraded signal and predicted one then set up a threshold to judge if we should replace the data point with interpolation. After the correction, we will get a much more smooth signal and remove the clicks in the degraded signal.

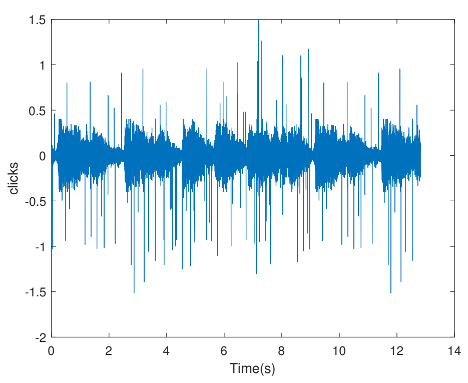

Figure 1: *clicks in degraded signal*

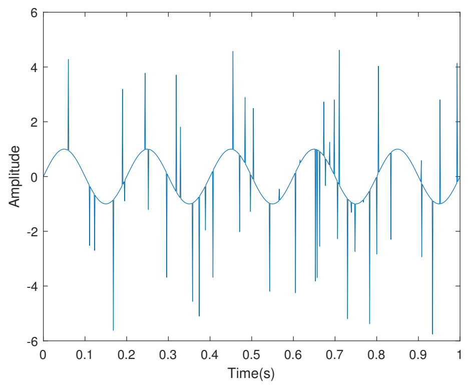

Figure 2: *generated clicks*

1.1 Detection

We assume that we could detect the noise in the voice by using AR model. Hence we generate the AR model to predict the voice waveform and find the residual between original signal between them. With the error we found, we could set a threshold to judge if the data point is not reliable.

1.2 Interpolation

Having discovered b_k we now need to replace these signal with the prediction. Again following our lectures we can use the AR model to interpolate the missing samples. Given $\mathbf{a}$ is our vector of AR coefficients and $\mathbf{b_k}$ is our detection vector and $\mathbf{s_o}$ is our original signal. We can insert the prediction data of the corresponding points into the distorted data and get a new data waveform.

2 The algorithm

The mathematical framework above only applies if the signal is statistically homogeneous. real signals are not, so we instead process the signal in blocks. Our algorithm is then as below and the flow chart as figure 3.

1. For each analysis block we process the inner N_a samples. The analysis block contains a prediction error block and overlap samples. The overlap samples could make the result blocks connected with each other more smoothly.

2. Normalise the block by removing the dc level by making the mean value to 0.
3. Estimate the AR coefficients of each analysis block. We can compute the coefficients of the AR model by using Normal Equations 2.

$$a = R^{-1}r \tag{2}$$

4. The residual of each sample e_k is simply the difference between the actual signal sample y_k and the prediction $\hat{y_k}$. Hence we can write 3.

$$e_k = y_k - \hat{y_k} = y_k - \sum_{p=1}^{P} a_p y_{k-p} = Ay = A_k y_k + A_u y_u \tag{3}$$

5. Threshold that error to give a detection indicator $b_k = (|e_k| > T)$ in each error block.
6. With b_k, we want to interpolate predition value into the error location to correct the waveform. We can assume that the error 3 are 0 and hence solve for the missing data as follows 4.

$$y_u = -[A_u^{'} A_u]^{-1} A_u^{'} A_k y_k \tag{4}$$

7. Replace the missing data samples with these prediction value and assembly these data block together, finally we can get the final corrected signal.

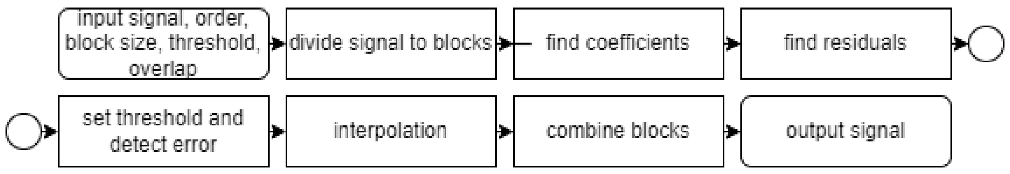

Figure 3: *block-based processing*

3 Experiments and results

3.1 Test with degraded artificial signal

An artificial signal was generated using gaussian distributed random numbers filtered with an IIR filter which order is 3. Then the signal was artificially degraded by random clicks. Figure 4 shows the ROC detection performance for different levels of degradation on this signal. We draw the scatterplot of TPR and FPR and fit. We found that the TPR/FPR level will increase with degradation level. As can be seen Figure 5 shows that less level of noise, the model will perform better.

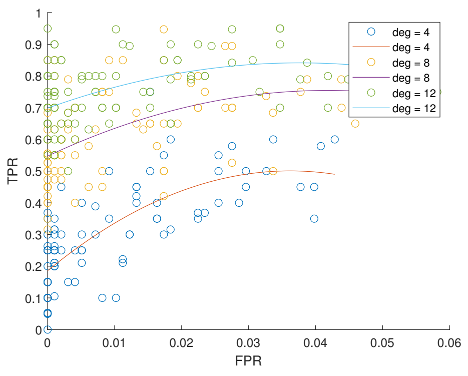

Figure 4: *TPR VS FPR(deg)*

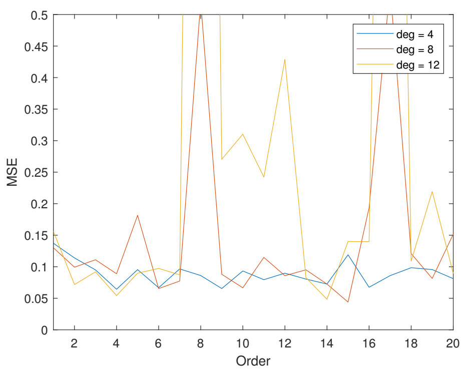

Figure 5: *MSE VS order(deg)*

3.2 Test with degraded clean signal

A clean signal was then degraded using random amplitude and random location click noise. Figure 6 and 7 shows ROC results of the model with different p and bs. Figure 8 shows the interpolation performance. We can find that higher order, block size will not lead to better performance absolutely. For example, TPR reaches to max value using p = 5. Thus we need to find out best parameters by assessing the performance of the model. Figure 9 shows the output signal and the clicks are removed efficiently.

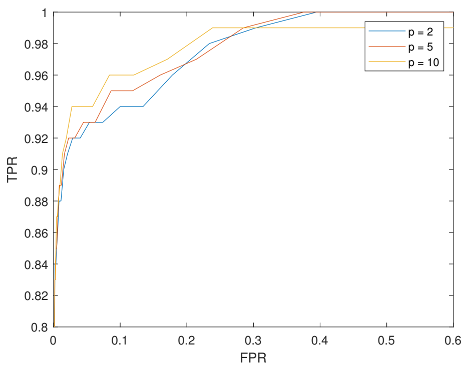

Figure 6: *TPR VS FPR (p)*

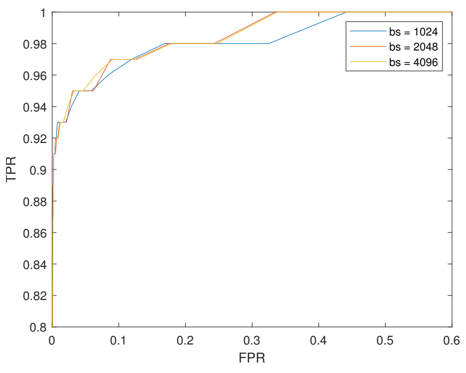

Figure 7: *TPR VS FPR (bs)*

3.3 Test with real signal

To test on a real signal the original music was processed with given parameters and get results shown in table 1. The cleaned up signal is saved as restored.wav.
According to the method in previous, when the order is 15, we get max TPR/FPR, but inappropriate order will lead to the increase of max error. When the block size is small, there will be more error. When the block size is larger than 1000, its performance is basically same, but it will have a

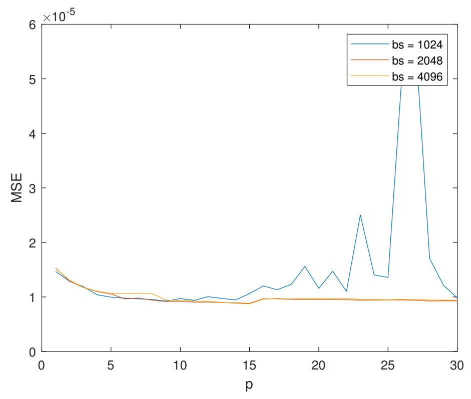

Figure 8: *MSE VS p(bs)*

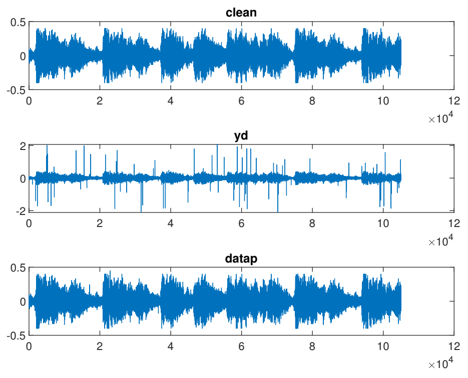

Figure 9: *compare signal*

impact on the running time.
Threshold will affect the detection performance. If it is small, the model will over predict and deviate from the original signal; if it is too large, it will fail to find the point that should be corrected.
Proper overlap can reduce the max error and TPR/FPR.
According to the above methods, we could try to find out best parameters by changing the parameters : order(1 - 30), block size(128 - 4096), threshold(0.1 - 5.0), overlap(0 - 50). The figure of exploring the relationship between parameters and performance is similar to the previous section. Due to space limitation, it will not be repeated here.
Observing the performance results and finally we obtained the parameters as shown in Table 1.

Table 1: *parameters and results*

parameter	order	block size	threshold	overlap	b_k num	data size
value	3	1200	0.35	2	134	105000
result	MSE	FPR(%)	TPR(%)	max error	TPR/FPR	run time(s)
value	0.0047	0.1	100	0.208	970.98	1.218

4 Conclusions

This thing actually worked. But it was really hard to find best parameter for a model considering the efficiency, detection and interpolation performance. I wish we could have finished the performance assessment function first then it will save me much time. The interpolation function still have some problems that if we get very small coefficients in some cases, due to inverse operation in equation 4, it will lead y_u to a large, impossible value. Maybe we could try other methods to acheive the interpolation or utilize threshold to avoid this case.