Assignment 2: Speech Compression and Quantization
EN2300 Speech Signal Processing

2011-10-23

Instructions for the deliverables: Perform all (or as many as you can) of the tasks in this project assignment. Summarize your results in a report style document. Explicitly state the problem numbers that you are addressing. The deliverable consists of your Matlab routines and your report. Both the Matlab code and the report should be unique for your group. Your Matlab code should be functioning and easy to read. Use a naming convention, which will allow us to associate easily a function name with a specific problem number. Running the codes should not depend on changing manually parameter values inside the functions, i.e. we should be able to run your code and see the result without having to change any source code. If there is some problem, we may ask you to come and present your work orally.

Grading: Each correctly solved task brings you a certain amount of points. This is indicated in the assignment text. The grade you get depends on the amount of points you accumulate. A passing grade corresponds to having 30 % and more of the total amount of points, while an excellent grade requires 85 % and more.

This instruction sheet, and more, is published on the course homepage, http://www.ee.kth.se/courses/EN2300/.

1 Introduction

On the course homepage you can find the speech data that you will analyze and compress in this assignment. Download the file `assignment2.mat`. Store it in a directory which is readable from the MATLAB prompt. To load the

data, type `load assignment2.mat` in MATLAB. Several MATLAB variables are then loaded into memory.

Throughout this assignment we will use the speech data in `speech8`[1] which is sampled at 8 kHz. `speech8` will be used both for tuning (training, optimizing) and evaluating the speech coders (evaluating on the training data is called closed test evaluation). Normally this is considered inappropriate (why?), but we accept it here for simplicity.

You will frequently be asked to evaluate the SNR which is calculated like $SNR = 10 \log_{10} \sigma_x^2/\sigma_q^2$, where $X(n)$ is input speech to the coder, and $q(n) = X(n) - \hat{X}(n)$ is the quantization error, where $\hat{X}(n)$ is the quantized speech. Thus, calculating the SNR amounts to estimating σ_x^2, and σ_q^2.

Please note that the last part of the assignment may be the most time consuming. There you are asked to design a forward gain and filter adaptive DPCM coder and you are not given many instructions. Make sure you allocate enough time for this last part!

2 The Uniform Scalar Quantizer

In this task we implement the simplest quantizer of all: the uniform scalar quantizer (USQ)[2]. USQs will be used frequently in this assignment.

TASKS/QUESTIONS

1. Implement a uniform scalar encoder with a function header
   `idx = sq_enc(in, n_bits, xmax, m)`
   where `in` is a vector with the original speech samples, `n_bits` is the number of bits available to quantize one sample in the quantizer (i.e., the rate), `xmax` and `m` define the range of the quantizer from `m-xmax` to `m+xmax`, so that the width of each quantization interval is $\Delta = 2 \times \texttt{xmax}/L$, where L is the number of quantization intervals and corresponding reconstruction values. `m` defines the mean (or offset) of the quantizer reconstruction levels. Setting `m` $= 0$ defines a "midrise" quantizer, and `m` $= \Delta/2$ gives a "midtread" quantizer (see the course book sec. 7.2). The function should return the index of the chosen quantization level.

   Implement the corresponding decoder function:

[1] Note that the amplitude is much greater than 1. To listen to the sound, you can use `soundsc` for automatic scaling, or else you must first scale the signal manually, before using `sound` or `wavplay`.

[2] Encoding and transmission by USQ is sometimes called Pulse Code Modulation (PCM).

`outq = sq_dec(idx, n_bits, xmax, m)`

where `outq` is the corresponding reconstruction value for `idx`.

The USQ is a highly structured quantizer. The encoder can be implemented essentially by only a scalar division (no multiplications, comparisons or loops are needed), making the computational complexity independent of the bitrate. Make sure your encoder has a computational complexity independent of the bitrate!

2. Run the encoder and decoder on a ramp signal `x=-6:0.01:6`. Use a 2-bit quantizer with $x_{max} = 4$. Plot the quantizer output as a function of the input. Make sure the output levels are exactly where you expect them to be. Use a quantizer mean `m=0`. Do a similar plot with `m=1.5`.

   (4 pts)

3 Parametric Coding of Speech

In this task we complete our design of the vocoder from assignment 1. Note that the speech used in this assignment is different from assignment 1 and if your vocoder analysis is not robust you may have to retune the analysis for this assignment. You are encouraged to use your own vocoder. However, it must produce a quality that is at least as high as our vocoder provided to you on the course web page. If not, then you have to get aquainted with our vocoder, and use that in the following.

In practice, a speech coder must operate on a frame-by-frame basis, and transmit compressed data as quickly as possible in order to keep the delay short. However, you may have implemented your vocoder in an off-line fashion (the pitch estimator is easier to implement like that for example). This is acceptable also here.

3.1 Quantizing the Gain

Here you design a quantizer for the gain parameter. Use a uniform scalar quantizer. Design it such that the overload distortion is negligible.

TASKS/QUESTIONS

1. Provide a plot of the histogram of the gain parameter. Indicate in the plot the range of the quantizer, i.e., mark the outer boundaries `m`$\pm$`xmax` (also mark `m`). Note that the pdf has a non-zero mean.

(1 pts)

2. Run the vocoder with a uniform scalar gain quantizer according to the design above. Find the rate at which you cannot hear the quantization distortion.

   (2 pts)

3. Take the logarithm of the gain parameter prior to quantization (does not matter which base). Provide a plot of the histogram of the gain parameter in the log-domain. Indicate the range of this quantizer as above.

   (1 pts)

4. Run the vocoder with a uniform scalar log-gain quantizer according to the design above. Find the rate at which you cannot hear the quantization distortion. Make sure to modify the decoder accordingly (apply the exp function to the quantized log-gain).

   (2 pts)

5. Which is better: gain quantization in linear or log domain?

   (2 pts)

3.2 Quantizing the Pitch and Voiced/Unvoiced Decision

Come up with an efficient way to encode the pitch and voiced/unvoiced decision!
(2 pts)

3.3 Quantizing the LP parameters

For the quantization of LP parameters, we will use a vector quantizer (VQ). You do not need to optimize (train) the VQs; that has been done for you, and the codebooks can be found in the MATLAB variables `lsfCB1` and `lsfCB2` in the file `assignment2.mat`. The codebooks constitute a multistage VQ. `lsfCB1` is a 10 bit VQ optimized on 10 dimensional LSF vectors. `lsfCB2` is a 10 bit second stage residual codebook. What you need to do is to program an encoding function and a corresponding decoding function for a multistage VQ. A suitable calling syntax for these functions can be
`codeA=encodefilter(A, cb1,cb2)` and
`Aq= decodefilter(codeA, cb1,cb2)`.
Here `A` is a matrix with filter coefficients stored row-wise, and `codeA` is a two-column matrix with the corresponding code indices, stored row-wise.

HINTS/REMARKS

1. To convert between polynomial (a-) coefficients and LSFs see `poly2lsf` and `lsf2poly`.

2. `poly2lsf` requires the polynomial coefficients to correspond to a minimum phase whitening filter. This is guaranteed by the autocorrelation LP analysis. `lsf2poly` requires that the LSFs correspond to a minimum phase whitening filter. The multistage VQ can output LSFs that do not satisfy this. As a precaution simply sort the LSFs prior to calling `lsf2poly`. Also check so they are between 0 and π.

(4 pts)

3.4 Optimizing the Bit Allocation

Here you experiment a little with the number of bits to use for each parameter. Since you are provided with the LSF codebooks, you cannot experiment with the number of bits spent on LP parameter quantization. LP quantization contributes a lot to the total number of bits, and the choice of LSF codebook size vs. performance is crucial in vocoder design, but here we keep the LSF codebook size fixed.

Find a bit allocation (i.e. the number of bits to use in each quantizer) for the gain, pitch, voiced/unvoiced quantizers, such that the quality is the same as when these parameters are unquantized (the effect of the LP parameter quantization is always present).

TASKS/QUESTIONS

1. Evaluate the SNR for your design above.

   (2 pts)

2. What number of bits do you suggest for the pitch? For the gain? For the voiced/unvoiced decision?

   (3 pts)

3. What is the rate in bits per sample of your vocoder with the bit allocation suggested above? In bits per second?

   (2 pts)

4. Does it make sense to evaluate SNR here? Why or why not?

   (2 pts)

4 Speech Waveform Quantization

4.1 Uniform Scalar Quantization of Speech

We will in the following design our quantizers using `xmax` $= k\sigma_X$, where σ_X^2 is the variance of speech and k is an experimentally tuned parameter (here it is tuned to maximize SNR). k is rate dependent and the SNR optimal k for rates 1 to 16 are

(0.95, 2.1, ?, 4.95, 6.3, 7.65, 8.85, 9.95, 10.6, 11.0, 11.1, 11.2, 11.15, 11.2, 11.15, 11.15)

The optimal (in terms of SNR) value for $R = 3$ you have to find experimentally. Run the quantizer with different values of k, and measure the SNR, to find the optimal choice.

TASKS/QUESTIONS

1. Evaluate the optimal k for $R = 3$.

   (2 pts)

2. Run the quantizer at rates 16, 15, 14, . . ., 2, 1, and evaluate the SNR for each rate. Provide a plot of the SNR as a function of rate.

   (2 pts)

3. Provide a graph of the theoretical SNR in the same plot as the experimental SNR plot. For the theoretical SNR, assume that the number of quantization levels is high, and that overload is negligible.

   (2 pts)

4. At what rate can you not tell the difference between the original and the quantized signal?

   (1 pts)

5. Listen to the quantization error signal, $q(n)$! How would you characterize $q(n)$ for a system operating at rate $R = 1$. Increase the rate (up to $R \approx 12$) and describe how the character of $q(n)$ changes.

   (1 pts)

6. OPTIONAL: Is it advantageous to have a reconstruction level in the origin for low rates? Compare (by listening) midrise and midtread quantizers at low bit rates.

   (2 bonus pts)

5 Adaptive Open-Loop DPCM

In this section we will study *open-loop* DPCM[3]. Open loop DPCM can be viewed as pre-filtering, quantization, and post-filtering, see Figure 1 (the adaptation mechanism is not shown). It is called open loop because the quantizer is not part of the prediction loop.

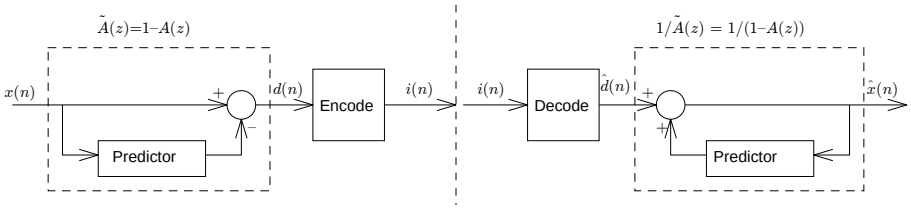

Figure 1: Open loop DPCM (adaptation not illustrated)

We will want to adapt both the LP coefficients and the gain in a forward fashion. See the vocoder section for hints on how to forward adapt those. In general the structure of open loop DPCM is similar to the vocoder structure with one important difference: vocoders create an artificial replica of the prediction error signal $d(n)$ (by estimating voiced/unvoiced, the pitch, energy), whereas open loop DPCM coders quantize each sample of the prediction error signal and thus try to preserve the original waveform. We recommend that you reuse as much as possible of the code from the vocoder.

TASKS/QUESTIONS

1. You are not given many guidelines here. Give it your best shot and make sure you can motivate your choice of for example

   - analysis frame length,
   - update length (to keep things simple make analysis and update lengths equal, i.e., no overlapping analysis frames),
   - window function (for the analysis of certain parameters),
   - number of bits to quantize the gain,
   - number of bits to quantize the residual,

   Use the VQ as before to quantize the LP parameters (thus, you need not decide prediction order!).

[3]Differential Pulse Code Modulation

Design the PCM quantizer for the prediction error $d(n)$ according to $x_{max} = k\sigma_d$. Optimize k for $R = 3$ (R meaning the rate of the residual quantizer), by experimenting, so that it sounds good, i.e., do not optimize SNR theoretically.

(4 pts)

2. Run your system at R = 3. How would you characterize the reconstructed speech? What does the quantization error sound like?

   (1 pts)

3. What shape does the quantization error spectrum have? Plot a DFT based spectrum of the error for a voiced frame. What does theory say?

   (2 pts)

4. Measure the SNR of your system. Compare with the SNR of PCM at the same rate. Comments?

   (2 pts)

5. What is the total rate of your coder in bits per sample? In bits per second?

   (2 pts)

6. Is it better to use the quantized LP coefficients in the encoder filter than to use the unquantized LP coefficients?

   (2 pts)