TSBK02 Computer Lesson: Quantization

April 2016

1 Introduction

The problems for this computer lesson deals with quantization (both scalar and vector quantization). We use Matlab to solve the problems. In order for Matlab to find the images and some of the functions, you must add the course directory to the search path:

```
>> addpath /site/edu/icg/tsbk02/matlab/
```

2 Distortion

We will measure the distortion as the mean-square error. Given an original signal `x` and a reconstructed signal `xhat` of any dimensionality, we can calculate the distortion using

```
>> D = mean((x(:)-xhat(:)).^2)
```

If the signals are onedimensional the `(:)` parts can be omitted.

3 Scalar Quantization

3.1 Uniform quantization

The simplest way to do uniform quantization is by dividing the signal to be quantized with the step size and then round to an integer. If we only want a limited number of intervals, the resulting integer sequence should then be thresholded, but this is not necessary for all applications. Reconstruction is done by multiplying the integer sequence with the step size. In Matlab, without thresholding, this can look like

```
>> q = round(x/delta);
>> xhat = q*delta;
```

given an input signal `x` and a step size `delta`. The signal `q` is an integer sequence suitable for source coding and `xhat` will be the reconstructed signal. Note that this gives a midtread quantizer, which is usually what we want.

For a limited number of quantization steps, try the function **uniquant**. The number of quantization levels given will determine the type of quantization. An odd number of levels gives a midtread quantizer, while an even number gives a midrise quantizer.

For fine uniform quantization, the distortion is approximately $D \approx \Delta^2/12$ where Δ is the step-size of the uniform quantizer. Check how good this approximation is by quantizing data (use the function **uniquant**) with different step-sizes and number of levels and calculate the resulting distortion. Suitable data to quantize can be random gaussian data (from the Matlab function **randn**). Try to determine for which combinations of stepsize and number of quantization levels the approximation is reasonable.

For each choice of the number of levels, determine which choice of step size that gives the lowest distortion.

3.2 Lloyd-Max quantization

The LBG algorithm is normally used to train vector quantizers, but it can of course also be used to find an optimal scalar quantizer (optimal in the sense that it gives the lowest distortion for a given number of reconstruction points), ie, it can be used to find Lloyd-Max quantizers. Train scalar quantizers for gaussian and laplacian noise, with different number of reconstruction points, and see how well they correspond to the Lloyd-Max quantizers that are listed in your formula collection.

Here's a small example on how it could be done for 4 quantization values.

Create a training set of 100000 one-dimensional gaussian values

```
>> train=randn(1, 100000);
```

Construct a random starting codebook with 4 one-dimensional vectors

```
>> cb=randn(1, 4);
```

Run the LBG algorithm for 50 iterations to find a good quantizer

```
>> cb=lbg(train, cb, 50);
```

Display the reconstruction values of the quantizer, sorted

```
>> sort(cb)
```

Create a new data signal

```
>> x=randn(1, 100000);
```

Quantize the data with the quantizer

```
>> xhat=nearest(cb, x);
```

What is the distortion?

```
>> D=mean((x-xhat).^2)
```

Alternatively the interactive function `trainvq` can be used

```
>> trainvq(train, 4)
```

Change **randn** to **randl** to get a Laplace distribution instead.

4 Vector Quantization

4.1 Vector quantization of noise data

Train vector quantizers for gaussian noise and compare to your result from 3.2. How much does the distortion for a given bitrate decrease when you use vector quantizers of higher dimensions? Remember that the number of reconstruction levels M at rate R and number of dimensions L is given by $M = 2^{RL}$. Note that L and RL should be integers.

An example would be similar to the example in 3.2, only with more dimensions. Note that for high dimensions the training and quantization might be very slow.

You can also try to quantize a memory source. A simple AR(1) process can be constructed by filtering gaussian noise like this

```
>> train=filter(1, [1 -0.9], randn(1, 100000));
```

The function **reshape** can be used to turn **train** into vectors for quantization.

For two-dimensional quantizers you can plot the decision regions, either by using the function **voronoi** or by clicking the "View Codebook" button in the **trainvq** window.

4.2 Vector quantization of images

Train vector quantizers for images. Use the **imread** function to load an image. You also need to use **im2col** to convert the image to vector format, and **col2im** to convert back to the image format. Compare vector quantizers of different dimensions (including scalar quantization). Suitable data rates are 1 bit/pixel or lower.

We give an example to show how it is done. The example coder uses 4-dimensional vectors and codes to a bitrate of 1 bit/pixel, ie the number of codebook vectors should be $2^{1 \cdot 4} = 16$.

Read an image, put it in the variable b

```
>> b=double(imread('boat.png'));
```

Display the image

```
>> imshow(b, [0 255])
```

Form 4-dimensional vectors from small 2×2 blocks in the image

```
>> bb=im2col(b, [2 2], 'distinct');
```

Construct a random starting codebook with 16 4-dimensional vectors

```
>> cb=rand(4, 16)*256;
```

Run the LBG algorithm for 50 iterations to find a good codebook

```
>> cb=round(lbg(bb, cb, 50));
```

(You can of course also use **trainvq** here to do the LBG algorithm.)
Quantize the image vectors using the codebook

```
>> bbq=nearest(cb, bb);
```

Rebuild an image from the quantized vectors

```
>> bq=col2im(bbq, [2 2], [512 512], 'distinct');
```

Display the quantized image

```
>> figure
>> imshow(bq, [0 255])
```
Calculate the distortion
```
>> D=mean((b(:)-bq(:)).^2)
```
Calculate the *peak-to-peak signal to noise ratio (PSNR)*
```
>> psnr=10*log10(255^2/D)
```

You could also try to quantize an image using a codebook that has been trained from another image to see how much the distortion and image quality is affected.

5 Useful Matlab functions

This is just a short list of Matlab functions that might be useful for solving the problems in this paper. All functions have online helptexts, just type `help <function>` at the Matlab prompt to read it.

Function	Short description
rand	Uniformly distributed random numbers
randn	Gaussian random numbers
randl	Laplacian random numbers
uniquant	Uniform quantizer
nearest	General vector quantizer
lbg	Train a vector quantizer codebook
trainvq	Interactive training of a vector quantizer codebook
voronoi	Plot a 2-dimensional codebook and its decision regions
imread	Read an image file
imshow	Display an image
im2col	Convert an image to a format suitable for vector quantizing
col2im	Convert column vectors back into an image

6 Images

The following images can be used

File name	Short description
goldhill.png	A photo of a small village
baboon.png	Closeup of a mandrill face
peppers.png	Tasty vegetables
boat.png	Fishing boats