Exercise Multimedia Technology
WS 2003/2004

Sheet 6 (November 28th, 2003)

Exercise 6.1 Linear Predictive Coding (part 2)

On the last sheet we did LPC based on examples in which a signal was constructed such that it was possible to predict the signal with no residual. This time a sample should be LP coded. This involves a least-square optimization in which the coefficients are chosen in order to minimize the difference between the prediction signal and the original sample.

The optimization is often done by means of a singular value decomposition (remember Analysis I). There is a very handy public domain library, the so-called GNU scientific library (GSL for short), which includes a vast number of optimization functions that have been improved and extensively tested over the years. The GSL will probably satisfy all the computational needs you will ever have in your working life.

Use the GSL or any other library to LP code the sample provided below. You can code small chunks (e. g., 5-100 samples) of the signal since approximating the whole signal in one pass is computationally not feasible. Vary the size of the data chunks and the number of coefficients which approximate them. What characteristics do the approximations have?

The sample is provided in the .au format, a very simple raw data format. A very brief description of the .au format is also provided below. You can play it on Linux using the play command: on windows the media player should handle the playback by default. Use your own samples if you like.

Hint: In the upcoming exam at the end of the term you can use the SGL function call (or an equivalent library call) and simply assume that it will do all the computation for you. If you have no experience with a numerical library, you will also be allowed to perform the singular value decomposition on your own.

References

Homepage of the GSL: http://www.gnu.org/software/gsl/

Manual: http://www.gnu.org/software/gsl/manual/gsl-ref_toc.html

If you don't want to read too much, this page probably includes all you need to know with an additional example: http://www.gnu.org/software/gsl/manual/gsl-ref_35.html#SEC460

Description of the .au format: http://www.informatik.uni-mannheim.de/informatik/pi4/data/au.txt

Sample signal 1: http://www.informatik.uni-mannheim.de/informatik/pi4/data/sample_mono_1.au

Sample signal 2: http://www.informatik.uni-mannheim.de/informatik/pi4/data/sample_mono_2.au

Allocate everything that is necessary for the best-fit:

```
gsl_matrix *X = gsl_matrix_alloc(no_constraints, no_parameters);
gsl_matrix *cov = gsl_matrix_alloc(no_parameters, no_parameters);
gsl_vector *y = gsl_vector_alloc(no_constraints);
gsl_vector *c = gsl_vector_alloc(no_parameters);

gsl_multifit_linear_workspace* p_workspace =
       gsl_multifit_linear_alloc(no_constraints, no_parameters);
```

define constraints:

```
for(int constraint = 0; constraint < no_constraints; constraint++) {
    for(int parameter = 0; parameter < no_parameters; parameter++) {
        gsl_matrix_set(X, constraint, parameter, sample[constraint+parameter]);
    } // for

    gsl_vector_set(y, constraint, sample[constraint+no_parameters]);
} // for
```

find best fit:

```
gsl_multifit_linear(X, y, c, cov, &chisq, p_workspace);
```

do the prediction:

```
for(int j = no_parameters; j < data_chunk; j++) {
    double prediction = 0.0;

    for(int parameter = 1; parameter <= no_parameters; parameter++) {
        prediction += sample[j - parameter] *
                      gsl_vector_get(c, no_parameters - parameter);
    } // for

    reconstructed_sample[j] = prediction;
} // for
```

Free what you have been using:

```
gsl_multifit_linear_free(p_workspace);

gsl_vector_free(c);
gsl_vector_free(y);
gsl_matrix_free(cov);
```

```
gsl_matrix_free(X);
```

The solution as a package is available at: **www.informatik.uni-mannheim.de/pi4/data/lpc.tgz**

Exercise 6.2.1 Quality of Service parameters

In reference to both networks and endsystems, what kinds of parameters did you get to know that influence QoS (find at least 6 of them) ?

Within a network: bandwidth, delay, delay jitter, loss, ...

Within a computer: computational capacity, realtime ability of the OS, memory

Exercise 6.2.2 Quality of Service artifacts

Describe the effect of unmatched quality contraints on a video stream. What kinds of artifacts does each parameter cause? Suggest solutions to solve or lessen those problems.

- Delay causes a retarded start of the video stream. A high delay also makes retransmission of essential data more difficult. Solution: Start playback later than requested by the user. If the data stream is transmitted more than once, a buffer could be built up in advance. If the user requests the playback of a video the stream is fed from the buffer while more data is requested over the network. This solution is not suitable for video conferencing.
- Delay jitter causes some packets to arrive too late to be included into the video decoding since the video might have advanced too far in time. Solution: Build up a buffer and start playback later. Small delays are acceptable for video conferencing as well (< 100ms). Larger delays will force the participants to use a protocol, indicating when one party has finished its talk.
- The same is true for insufficient bandwidth. A potential solution would be to adapt the data rate. A lower resolution or less sharpness of the image is more acceptable for the eye than the loss of entire frames or blocks. Continuity should be favoured over details or sharpness.
- Insufficient computational capacity means that not all frames can be processed in time. Usually a decoder will only be able to drop entire frames in order to resynchronize itself to the actual time of the movie. Slowing down the playback is not recommended since at least the audio should be synchronous. Dropped frames can also result from insufficient realtime capabilities of the operating system.
- Data that has actually been transmitted might not fit into the memory. The effect is equivalent to packet loss. In order not to make matters worse the oldest packet should be dropped first.

Exercise 6.2.3 Quality of Service artifacts

Have a look at the slides (the pdf-file) of the 2001 Keynote Talk at the annual Workshop IWQoS and despite the massive use of acronyms, try to guess what it is about.

(1) Why isn't there something like QoS on the internet after so many years of research?

- Not too many users require QoS

- Bandwidth is cheap, so increasing bandwidth will also eliminate problems from exercise 6.2.2
- QoS requires more „intelligence“ in routers. The processing of packets significantly reduces a router's capacity. In the worst case this would introduce QoS problems that would not exist with simple but fast routers that do not do QoS.
- To be efficient, QoS would have to be supported by the whole network (worldwide if possible). Therefore end-users, service providers and backbone providers would have to cooperate. This is unlikely.
- IP was so successful because it is at the same time simple and robust under many conditions.

(2) Do you think that QoS is worth the effort and will it ever catch on?

No default solution is expected here.

http://www.cs.columbia.edu/~hgs/papers/Schu0106_Quality.pdf