Audio Data

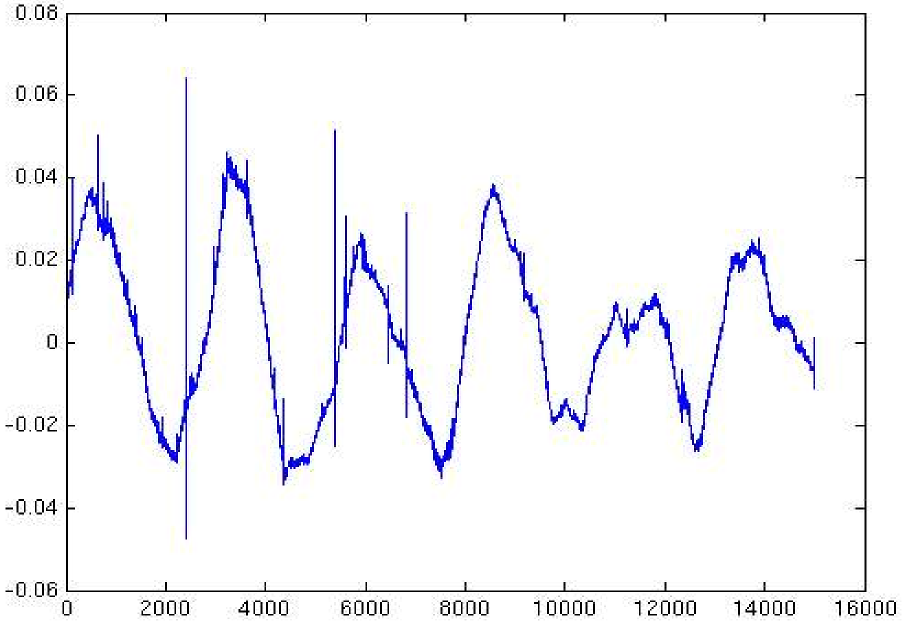
- Lots of audio files in the world
  - Home recordings
  - Recordings of concerts
  - BBC archive data
  - ...

Damaged audio data

- Audio files may be damaged
  - Clicks
  - Hiss
  - Clipping
  - Missing data
  - ...
- So, estimate the original data: P(original | data)

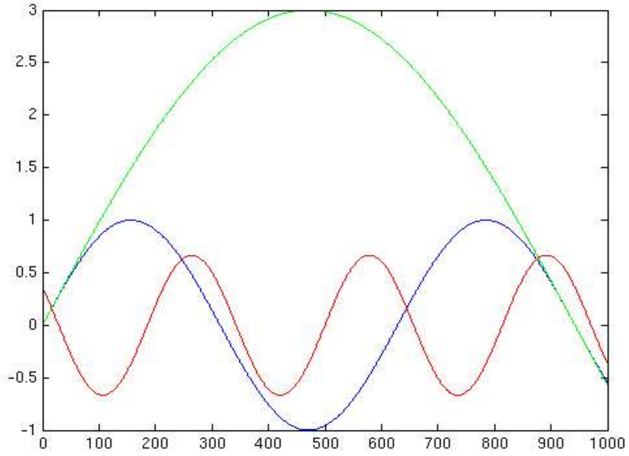
Frequency or time?

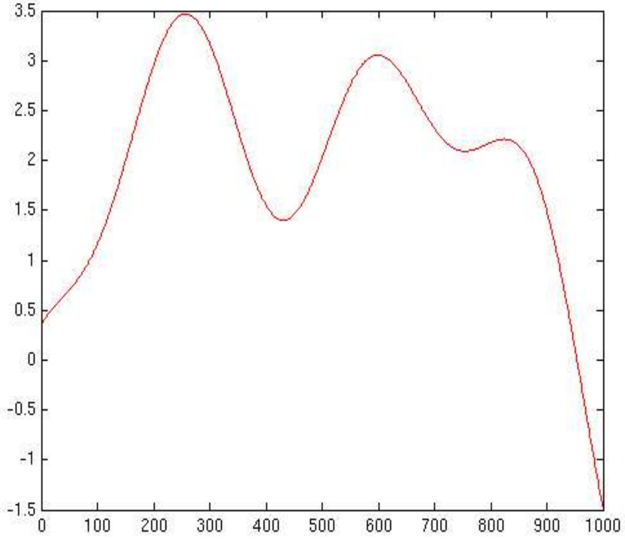
- Can view an audio file as a time sequence



Frequency or time?

- Or as a sum of frequency information





- Describe with amplitude for each frequency.

Fourier Transforms

- The conversion from a time series to a frequency series is called a Fourier transform

$$F(u) = \int \left(f(x) * \exp(i u x) dx \right)$$

$$f(x) = \frac{1}{(2 pi)} \int \left(F(u) * \exp(-iux) dx \right)$$

(recall exp(ix) = cos(x) + isin(x))

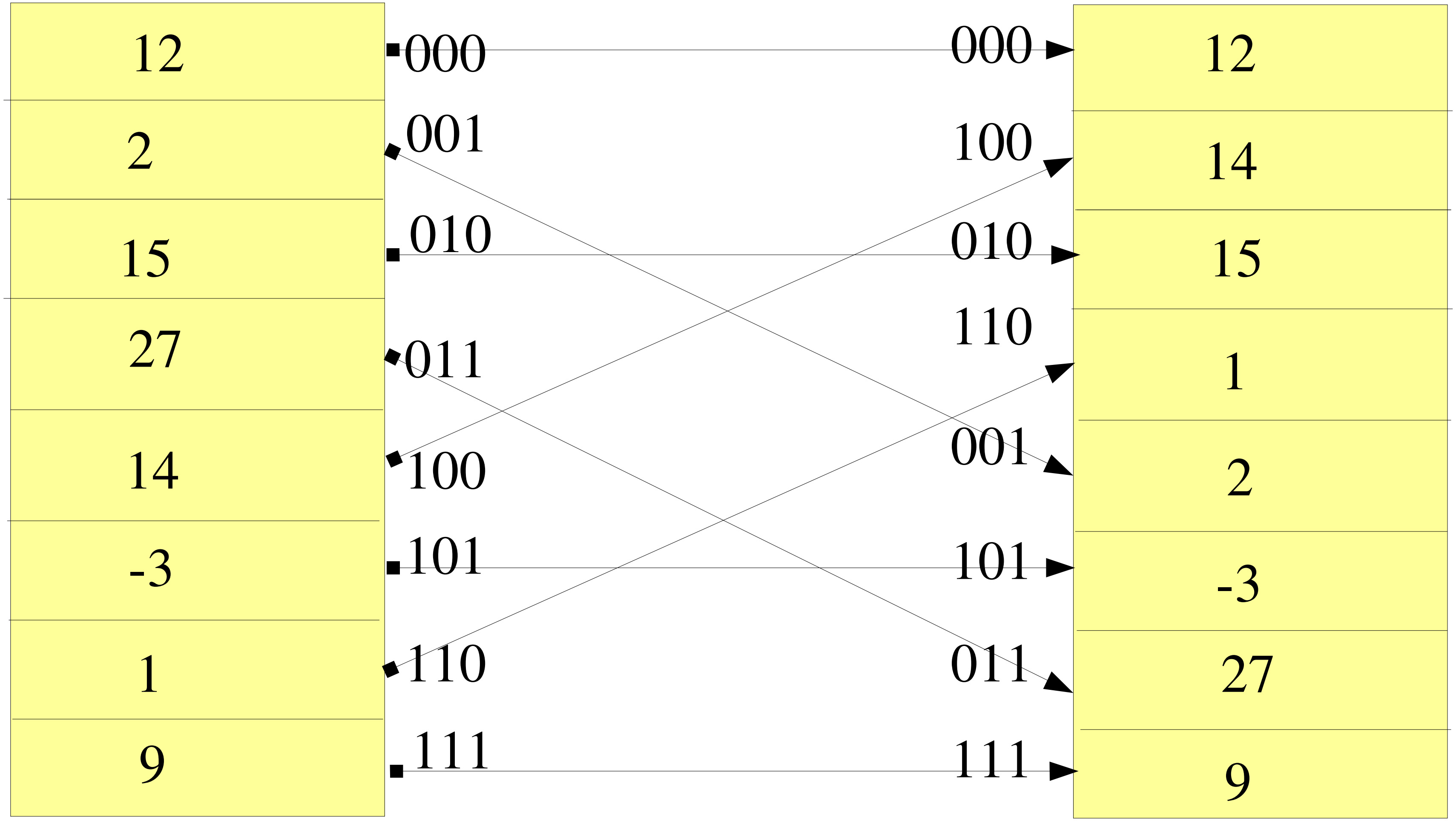
Fourier Transforms

- On finite data, a discrete Fourier transform: sum rather than an integral.
- Efficient algorithm for computing a discrete Fourier transform (the list of coefficients F(u)): fast Fourier transform (FFT).

Fast Fourier transform

- Fourier transform is the sum of an “odd” FT and an “even” FT
- Each of those can be divided again
- End up with a series of one point FTs:
  this turns out to be the data in reverse bit order.

Reverse bit order

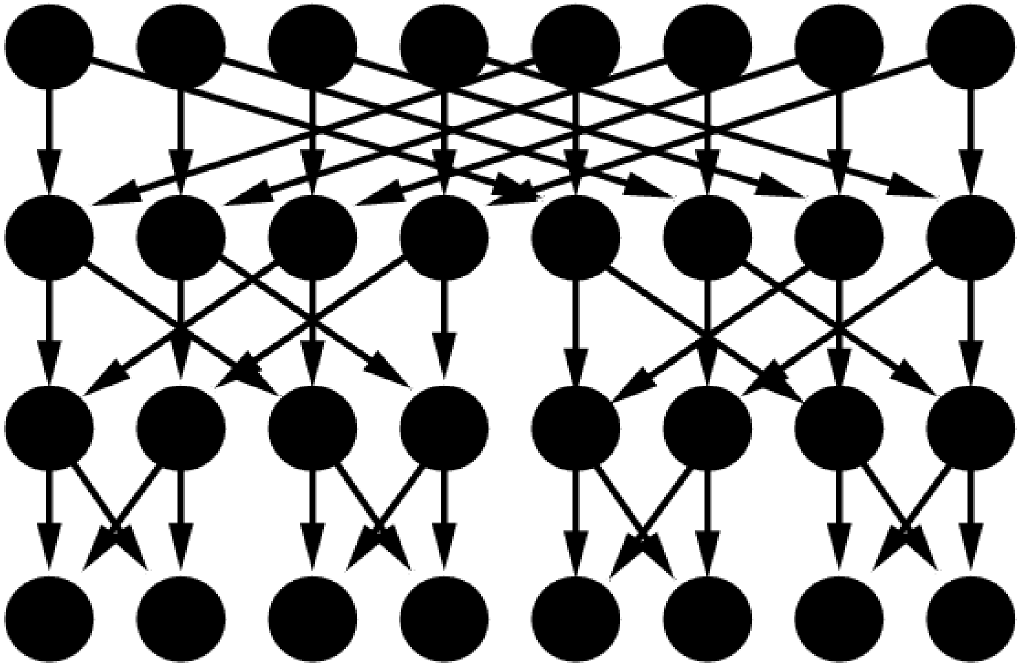


12	000	000	12
2	001	100	14
15	010	010	15
27	011	110	1
14	100	001	2
-3	101	101	-3
1	110	011	27
9	111	111	9

Fast Fourier Transform

- Can view the FFT as a network structure:

Fourier components



Data points

Belief propagation

- Techniques for finding the conditional probability at a node of a belief network (here, the Fourier components), given the prior probability and the observed data (here the time series data points)

Probabilistic FFT

- Method for restoring missing data:
  - Supply priors on Fourier components
  - Compute a probable FT transform given data which is present, using the network structure for belief propagation
  - Use inverse FT to estimate missing data

Implementation

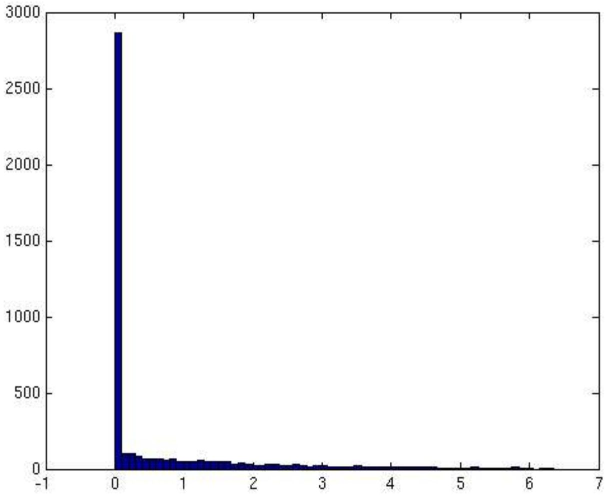
- Estimating FT given data:
  - Conjugate gradient method
  - Model is in form of gradient function for P(FT | known_data, unknown_data)
- Conjugate gradients, FFT implemented in C
- Octave interface for loading audio files, supplying data and gradient function

Beyond missing data

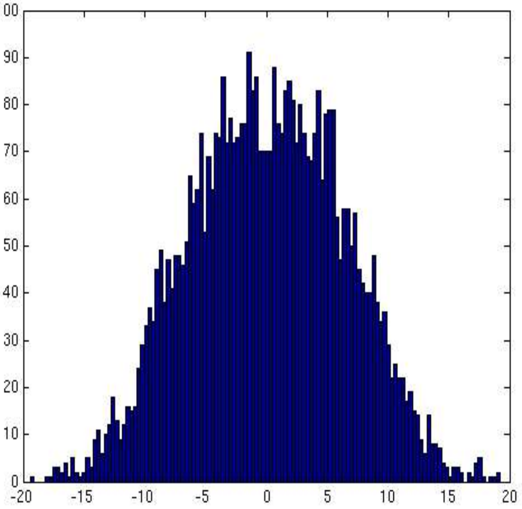
- Add a second layer: a model for the probability of the true data given the observed data
- Handle clicks (similar to missing data), hiss, clipping (large values likely to be clipped: broad range of probabilities for original data, smaller values likely to be accurate), ...

Models

- Where do we get the priors from?
  - Look at real data



10th component



First component

Done so far

- C code, octave interface to apply conjugate gradient method when supplied with data and gradient function
- Appears successful if gradient function is very simple (eg, Data – 4)
- Attempt with gradient of a Gaussian: gradients blow up to -inf. To debug...

Still to do

- Script to load audio files: octave doesn't have matlab's wavread()
  - Or, translate to matlab
- Add layer to estimate P(truth | data)
- Experiment with different models for Fourier components and P(truth | data)
- Report!