Lossy Compression

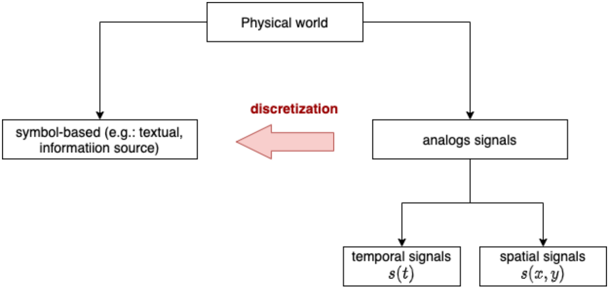
Sampling and quantization

October 2018

Content

- Analog signals
- Lossy compression: Distortion measures
- Convolution and Sampling

Analog signals and discretization



- Discretization helps convert analog signal to information source models
- We can apply coding/processing techniques in that domain

Analog signals: Time-domain view

Continuous signals that contain time-varying quantities

- Amplitude
- Frequency
- Phase

Examples:

- Sound wave
- Electo-magnetic wave
- Light signals

Time-domain view $s = f(t)$

Analog signal representation

- $s(t)$ is polynomial

$$s(t) \iff (a_k | k = 0..N)$$

- $s(t)$ is sinusoidal

$$s(t) \iff (A, f, \phi)$$

- Not a good way since it depends on the form of $s(t)$

Fourier Series

Any periodic function $s(t)$, with period T, can be decomposed into Fourier series. Two forms of Fourier representation

$$\begin{aligned} s(t) &= a_0 + \sum_{k=1}^{\infty} a_n \cos\left(\frac{2\pi k}{T}t\right) + \sum_{k=1}^{\infty} b_n \sin\left(\frac{2\pi k}{T}t\right) && (1) \\ &= \sum_{k=-\infty}^{\infty} c_n e^{i\frac{2\pi k}{T}t} && (2) \end{aligned}$$

Values of $\{a_k, b_k\}$, or c_k can be obtained by *Fourier transform*

$$\begin{aligned} a_k &= \frac{2}{T}\int_T s(t) cos(\frac{2\pi k}{T}t)dt && (3) \\ b_k &= \frac{2}{T}\int_T s(t) sin(\frac{2\pi k}{T}t)dt && (4) \\ c_k &= \frac{1}{2}(a_k - ib_k) = \frac{1}{T}\int_T s(t) e^{i\frac{2\pi k}{T}t} && (5) \end{aligned}$$

Fourier Series: Example

- Square wave:

$$square(t,1) = A\sum_{k=0}^{\infty}\frac{1}{2k+1}\sin(2\pi(2k+1)t) \tag{6}$$

- Sawtooth wave:

$$sawtooth(t,1) = A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt) \tag{7}$$

Analog signal representation

- $s(t)$ is polynomial

$$s(t) \Longleftrightarrow (a_k | k = 0..N)$$

- $s(t)$ is sinusoidal

$$s(t) \Longleftrightarrow (A, f, \phi)$$

- Frequency domain representation

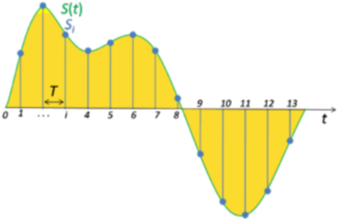
$$s(t) \Longleftrightarrow (c_k | k = -\infty...\infty)$$

Discretizing analog signals: sampling

Given an analog signal $s = f(t), t \in \mathbb{R}$
Calculate $s_{sample} = \{s_i = f(t_i) | t_i = iT, \forall i \in \mathbb{Z}\}$

T

- s_{sample}: sampled signal
- s_i : one sample of s_{sample}
- T : sampling period
- $f = \frac{1}{T}$: sampling rate, sampling frequency



Signals can be:

- Composed: $s_1(t) + s_2(t) \longrightarrow s(t)$
- Decomposed: $s(t) \longrightarrow s_1(t) + s_2(t)$

Sampling theorem

A band limited signal can be exactly reconstructed if it is sampled at a rate at least twice the maximum frequency component in it

- Nyquist frequence: $f_{nyquist} = 2 \times f_{max}$
- Inadequate frequence: $f_{sample} < 2 \times f_{max} \longrightarrow$ Aliasing
- Oversampling frequence: $f_{sample} > 2 \times f_{max}$

How to avoid aliasing:

- Oversampling
- Filter hight frequency (using a low-pass filter) then sampling

Reconstrucing signal from samples

Input: A sampled signal $x_n, n = 1..N$
Output: Reconstruct $x(t)$ at any t

$$x(t) = \sum_{n=-\infty}^{\infty} x_n sinc\left(\frac{\pi}{T}(t - nT)\right) \quad (8)$$

$$= \sum_{n=-\infty}^{\infty} x_n \frac{\sin\left(\frac{\pi}{T}(t - nT)\right)}{\frac{\pi}{T}(t - nT)} \quad (9)$$

Analog signal representation

- ...
- Frequency domain representation

$$s(t) \Longleftrightarrow (c_k | k = -\infty ... \infty)$$

- Sampled representation

$$s(t) \Longleftrightarrow (s_i | i = 0 ... N)$$

- Sampled representation in frequency domain : DFT

$$s(t) \Longleftrightarrow (s_i | i = 0 ... N) \Longleftrightarrow (F_i | i = 0 ... N)$$

Discrete Fourier Transform

We are more interested in discrete version. Discrete Fourier Transform:

- See signal as a series of samples in time domain: $s_i = f(iT_s), i = 1..N$
- See signal as a series of frequencies in frequency domain: $c_k = F\left(k\frac{2\pi}{T_s}\right), k = 1..N$

$$c_k = \frac{1}{2}(a_k - jb_k) = \frac{1}{N}\sum_{n=0}^{N-1} f\left(\frac{n}{N}T\right)e^{j\frac{2\pi kn}{N}} \tag{10}$$

$$a_k = 2Re(c_k); b_k = -2Img(c_k) \tag{11}$$

Fast Fourier Transform is an algorithm that perform DFT efficiently

Fourier Transform characteristics

- Energy preservation (Parseval theorem)

$$\int_{-\infty}^{\infty} |f(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(\omega)|^2 d\omega \tag{12}$$

- Modulation property

$$FT[f(t)cos(\omega_0 t)] = \frac{1}{2}\left(F(\omega - \omega_0) + F(\omega + \omega_0)\right) \tag{13}$$

- Convolution theorem

$$f(t) = f_1(t) \oplus f_2(t) \iff F(\omega) = F_1(\omega)F_2(\omega) \tag{14}$$

Typical discrete data: 1D temporal data

- G711 audio data:
  - Used by telephone system
  - Sampling rate: $8kHz$
  - 8-bit per sample
- CD-DA audio data:
  - Music CD application
  - Sampling rate $44100Hz$
  - 16-bit per sample
  - Stereo: 2 channels

Typical discrete data: 2D spartial data (raw image)

Matrix of pixels

- Binary image:
  - 1-bit per pixel
- Grayscale image:
  - 8-bit per pixel (luminance only)
- Color images
  - Multiple channels (3 or 4)
  - Various color modes exist: RGB, RGBA, YCrCb, HSL
  - Various number of bits per color component

Lossless vs. Lossy compression

- Lossless compressions produce codes whose $L(C, X) \geq H(X)$ $\Longrightarrow$ Information content is preserved (lossless)
- Lossy compressions may cause $L(C, X) < H(X) \Longrightarrow$ Accept information loss!!
- Examples: ...
  - Remove some "unimportant" source symbol set (Compress english text)
  - Use least significant bit for marking a pixel (Run length encoding a gray image)
- How to compress data depending on characteristics of data

Fidelity and distortion

Fidelity: how similar to original
Distortion: how different from original

Assume that source outputs are numeric (called signals/samples).
How???

- Source output: $\{x_i\}$. Reconstructed sequences: $\{y_i\}$
- Difference distortion measures:
  - Squared Error: $d(x, y) = (x - y)^2$
  - Absolute Error: $d(x, y) = |x - y|$
- MSE:

$$\sigma_X^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2$$

Practical measures

- Average source output: S_X
- Signal-to-noise ratio (SNR)

$$SNR = \frac{S_X^2}{\sigma_X^2}$$

- Decibels (dB) SNR:

$$SNR(dB) = 10\log_{10} SNR$$

- Peak SNR:

$$PSNR(dB) = 10\log_{10} \frac{\max_i\{x_i^2\}}{\sigma_X^2}$$

- Maximum error:

$$d_\infty = \max_i |x_i - y_i|$$

Techniques for lossy compression

- Approaches for lossy compression
  - Drop some samples
  - Drop some least significant bits from source symbols (signals, or samples)
- Resulting sequences can have smaller *rate* but has *distortion*
- Rate-distortion function $R(D)$: The lowest rate at which the source can be encoded while keeping the distortion lower than or equal to D
- Human Audio/Visual System
  - Visual: Has limited spatial resolution
  - Allow signals in finite range. Audio: $20\ Hz$ to $20\ kHz$. Visual: λ_{red} to λ_{violet}
  - Other characteristics

Convolution

- Continous version

$$f_1(t) \oplus f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t-\tau) d\tau \quad (15)$$

$$= \int_{-\infty}^{\infty} f_1(t-\tau) f_2(\tau) d\tau \quad (16)$$

- Discrete version

$$z = (x \oplus y)[n] = \sum_{m=-\infty}^{\infty} x[m] y[n-m] \quad (17)$$

Convolution:linear

$$\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$$

$$\begin{pmatrix} z_0 \\ z_1 \\ \dots \\ z_m \\ \dots \\ z_{n+m-1} \end{pmatrix} = \begin{pmatrix} y_0 & 0 & \dots & 0 & \dots & 0 \\ y_1 & y_0 & \dots & 0 & \dots & 0 \\ y_2 & y_1 & y_0 & \dots & \dots & 0 \\ \dots & & & & & \\ y_m & y_{m-1} & \dots & y_0 & \dots & 0 \\ \dots & & & & & \\ 0 & \dots & y_m & y_{m-1} & \dots & y_0 \\ \dots & & & & & \\ 0 & \dots & 0 & \dots & 0 & y_m \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{pmatrix}$$

Convolution: Cyclic

$$
\begin{aligned}
\mathbf{z} &= \mathbf{x} \oplus \mathbf{y} \\
\begin{pmatrix} z_0 \\ z_1 \\ \ldots \\ z_n \end{pmatrix} &= \begin{pmatrix} y_0 & 0 & \ldots & 0 & y_2 & y_1 \\ y_1 & y_0 & \ldots & \ldots & 0 & y_2 \\ y_2 & y_1 & y_0 & \ldots & \ldots & 0 \\ 0 & y_2 & y_1 & y_0 & \ldots & 0 \\ \ldots & & & & & \\ 0 & \ldots & 0 & y_2 & y_1 & y_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \ldots \\ x_n \end{pmatrix}
\end{aligned}
$$

For cyclic convolution:

$$\mathbf{z} = \mathbf{x} \oplus \mathbf{y} \Leftrightarrow DFT(\mathbf{z}) = DFT(\mathbf{x})DFT(\mathbf{y})$$

Cyclic vs. Linear Convolution

Calculate cyclic convolution from linear convolution
Input: signal x and kernel h. $len(h) < len(x)$
Output: cyclic convolved signal z

1. Padding h with zeros up to $len(x)$
2. Linear convolve x with hh (or xx with h to get lz
3. Skip drop first $len(x)$ items and $len(x) - 1$ last items from lz to get z

Example:

Signal filters

Filtering: Convole sample signal with a *kernel vector* $\longrightarrow$ cause frequency content changed

Example: calculate $\mathbf{x} \oplus \mathbf{h}$

$$\begin{aligned} \mathbf{x} &= [1, 0, 0, 1, 8, 2] \\ \mathbf{h} &= [1, 0, 1] \end{aligned}$$

Convolution: applications

- Smoothing filters
- Pattern recognition
- Image processing