Lecture 25
FFT

David Semeraro

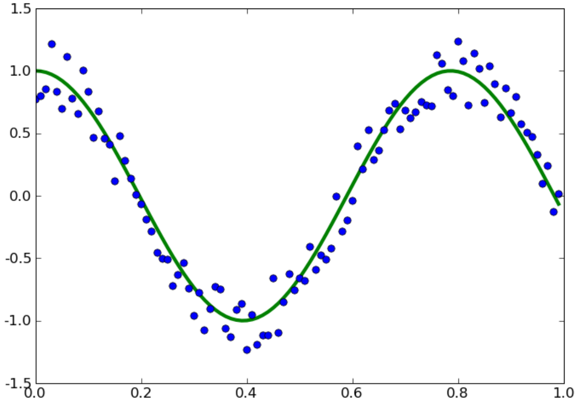
University of Illinois at Urbana-Champaign

Some slides by M. Heath

December 3, 2013

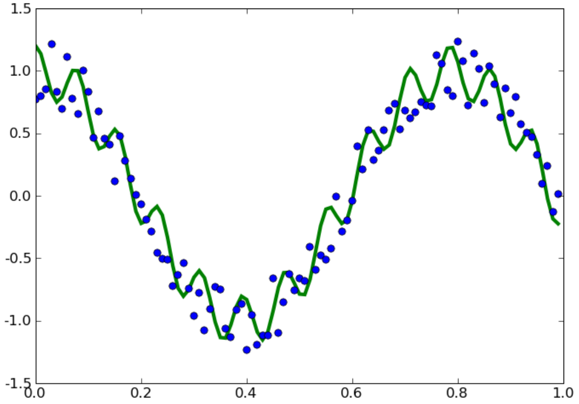
Problem

- Given some data:
  - What is the nature of the data?
  - What is the period?
  - Can we rid the data of noise?



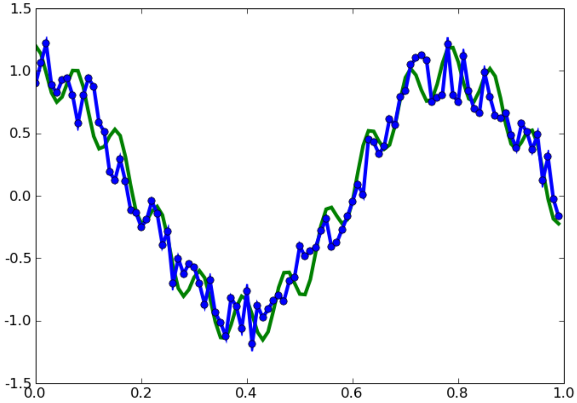
Problem

- Given some data:
  - What is the nature of the data?
  - What is the period?
  - Can we rid the data of noise?



Problem

- Given some data:
  - What is the nature of the data?
  - What is the period?
  - Can we rid the data of noise?



Problem Examples

- Example: remove (high-frequency) noise in an input signal
- Example: Weather data often contain different cycles
  - daily data
  - yearly data
  - want to isolate one of these
- Example: Economic data needs seasonal adjustment
  - remove unwanted periodicities to reveal “secular” trends
- Example: digital filtering in audio

Other Applications

- Spectral Analysis
- Data compression
- Partial Differential Equations

Recall Least Squares

- Minimize the norm of the residual $\Phi(x) = ||r||_2^2$.
- Residual $r = b - Ax$ where $A \in R^{nxm}$
- Geometrical interpretation: b is projected onto $R(A)$. The vector r is orthogonal to $R(A)$.
- Construct orthonormal basis for $R(A)$ and project b onto it.
- $P_A = A(A^TA)^{-1}A^T$ is the orthogonal projector onto $R(A)$.
- Choose x to minimize the distance between b and the column space of A (key idea)

Vector space to function space

In the finite case we are constructing a vector $v \in R^n$ such that $b - v$ is minimized. In addition we can write v as:

$$v = x_0 a_0 + x_1 a_1 + \cdots + x_{m-1} a_{m-1}$$

where the a_i are the columns of A.
Instead of vectors now consider functions. We wish to find the "best" approximation to a function f by a linear combination of bais functions rather than vectors.

$$f^* = c_0 \phi_0 + c_1 \phi_1 + \cdots + c_n \phi_n$$

Vector space to function space

$$f^* = c_0\phi_0 + c_1\phi_1 + \cdots + c_n\phi_n = \sum_{i=0}^{n} c_i\phi_i$$

Here the ϕ_i are functions and are taken to be linearly independent ($\sum_{j=0}^{n} c_j\phi_j = 0$ implies $c_j = 0$ for all j). We choose the coefficients c_j such that f^* lies the shortest distance from f.
The shortes distance from a point to a linear subspace is the length of the vector, between the point and subspace, that is perpendicular to the subspace. (think of the subspace as a plane and the point as the function f)

$$||f - f^*||^2 = \int_a^b |f^*(x) - f(x)|^2\,dx$$

Inner products

The inner product is defined as follows:

- Continuous $(f, g) = \int_a^b f(x) g(x)\, dx$
- Discrete $(f, g) = \sum_{i=0}^{n} f(x_i) g(x_i)$

Properties of inner products:

- $(f, g) = (g, f)$
- $(f, f) \geqslant 0$
- $(\alpha f + \beta g, \phi) = \alpha(f, \phi) + \beta(g, \phi)$

Orthogonal systems

- Two functions f and g are said to be orthogonal if $(f, g) = 0$
- A sequence of functions $\phi_0, \phi_1, \cdots, \phi_n$ form an orthogonal system if $(\phi_i, \phi_j) = 0$ for $i \neq j$ and $\|\phi_i\| \neq 0$ for all i.
- If $\|\phi_i\| = 1$ for all i the system is orthonormal.

Function approximation by Least Squares

When $\phi_0, \phi_1, \cdots, \phi_n$ are linearly independent the least squares problem has a unique solution:

$$f^* = \sum_{j=0}^{n} c_j^* \phi_j$$

where the coefficients c_j^* satisfy the normal equations:

$$\sum_{j=0}^{n} (\phi_j, \phi_k) c_j^* = (f, \phi_k) \quad (k = 0, 1, 2, \cdots, n)$$

This follows from the orthogonality property that $f - f^*$ is orthogonal to all the ϕ_j.
If $\phi_0, \phi_1, \cdots, \phi_n$ form an orthogonal system the orthogonal (or Fourier) coefficients satisfy:

$$c_j^* = \frac{(f, \phi_j)}{(\phi_j, \phi_j)}$$

Fourier Series

We seek the Least-squares approximation of a continuous function f that is periodic on the interval $[-\pi, \pi]$ using trigonometric polynomials as a basis. Consider the orthonormal basis:

$$\begin{aligned} U &= \{\phi_0, \phi_1, \cdots, \phi_{2N-1}, \phi_{2N}\} \\ &= \left\{ \frac{1}{\sqrt{2\pi}}, \frac{1}{\sqrt{\pi}} cos(x), \frac{1}{\sqrt{\pi}} sin(x), \cdots, \frac{1}{\sqrt{\pi}} cos(Nx), \frac{1}{\sqrt{\pi}} sin(Nx) \right\} \end{aligned}$$

Since the basis functions are orthonormal ($(\phi_i, \phi_i) = 1$) the fourier coefficients become $c_j = (f, \phi_j)$. The projection of f onto the space spanned by the basis functions can be written:

$$f^* = (f, \phi_0)\phi_0 + (f, \phi_1)\phi_1 + \cdots + (f, \phi_{2N-1})\phi_{2N-1} + (f, \phi_{2N})\phi_{2N}$$

Fourier Series

Represent the inner product terms as coefficients:

$$\begin{aligned}
\frac{1}{2}a_0 &= \left(f, \frac{1}{\sqrt{2\pi}}\right)\frac{1}{\sqrt{2\pi}} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\,dx \\
a_n &= \left(f, \frac{1}{\sqrt{\pi}}cos(nx)\right)\frac{1}{\sqrt{\pi}} = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)cos(nx)\,dx \\
b_n &= \left(f, \frac{1}{\sqrt{\pi}}sin(nx)\right)\frac{1}{\sqrt{\pi}} = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)sin(nx)\,dx
\end{aligned}$$

We can now write f^* as:

$$f^* = \frac{1}{2}a_0 + a_1cos(x) + b_1sin(x) + a_2cos(2x) + b_2sin(2x) + \cdots + a_ncos(nx) + b_nsin(nx)$$

Fourier Series

The N^{th} order Fourier approximation to f is then:

$$f^* = \frac{1}{2}a_0 + \sum_{n=1}^{N} [a_n cos(nx) + b_n sin(nx)]$$

The accuracy of this approximation imporves as N gets larger. The Fourier Series of f on $[-\pi, \pi]$ is:

$$f^* = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} [a_n cos(nx) + b_n sin(nx)]$$

Interpolation by trig

- When we have periodic data, use periodic interpolating functions: sines, cosines, etc
- Decompose a function into a combination of sines and cosines (as we did with polynomial basis functions)
- Decomposing into (co)sines of different “frequencies” naturally orders the data.
- The gist: working in a frequency (transformed) space is easier (faster) than the time domain.

Recall Fourier

- Suppose $f(x)$ is periodic on $[-\pi, \pi]$, then the Fourier Series for $f(x)$ is

$$f(x) \approx \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(nx) + b_n \sin(nx)$$

  with

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\, dx \quad b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\, dx$$

- Example: $f(x) = x$ on $[-\pi, \pi]$ (sawtooth). Then

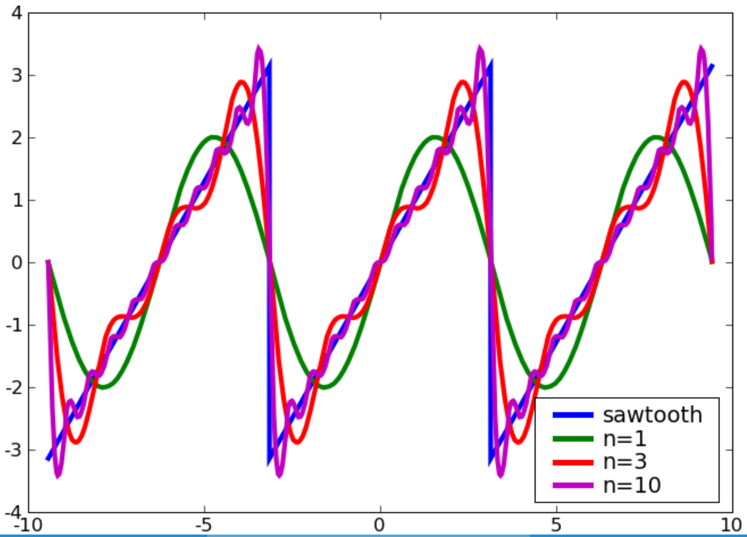
$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} x \cos(nx)\, dx = 0$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} x \sin(nx)\, dx = 2\frac{(-1)^{n+1}}{n}$$

  so

$$f(x) = 2 \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \sin(nx)$$

Sawtooth Fourier



General Fourier

- Use complex exponential notation using Euler's Identity:

$$e^{ix} = \cos(x) + i\sin(x), \qquad i = \sqrt{-1}$$

- Pure cosine and sine and be written with this:

$$\cos(x) = \frac{e^{ix} + e^{-ix}}{2}, \quad \sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$$

- Then a Fourier Series on interval $[a, b]$ with period τ is

$$g(x) = \sum_{n=-\infty}^{\infty} G_n e^{i2\pi nx/\tau}$$

with

$$G_n = \frac{1}{\tau}\int_a^b g(x)e^{-i2\pi nx/\tau}\,dx$$

Roots of Unity

- A primitive n^{th} root of unity for integer n is given by

$$\omega_n = \cos\left(\frac{2\pi}{n}\right) - i\sin\left(\frac{2\pi}{n}\right) = e^{-\frac{2\pi i}{n}}$$

- The n^{th} roots of unity are called twiddle factors here and are given by ω_n^k or ω_n^{-k}, $k = 0, \ldots, n-1$

Discrete Fourier Transforms

- The Discrete Fourier Transform (DFT) of sequence $x = [x_0, \dots, x_{n-1}]^T$ is a sequence $y = [y_0, \dots, y_{n-1}]^T$ given by

$$y_m = \sum_{k=0}^{n-1} x_k \omega_n^{mk}, \quad m = 0, 1, \dots, n-1$$

- in matrix notation this is

$$y = F_n x, \qquad \{F_n\}_{mk} = \omega_n^{mk}$$

- Example with $n = 4$:

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

Inverse Discrete Fourier Transforms

- Note:

$$\frac{1}{n}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3}\\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6}\\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9}\end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \omega^{1} & \omega^{2} & \omega^{3}\\ 1 & \omega^{2} & \omega^{4} & \omega^{6}\\ 1 & \omega^{3} & \omega^{6} & \omega^{9}\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

- So $F_n^{-1} = \frac{1}{n}F_n^*$.
- Thus the Inverse DFT is

$$x_k = \frac{1}{n}\sum_{m=0}^{n-1} y_m \omega_n^{-mk}, \quad k = 0, 1, \ldots, n-1$$

- Result: DFT, IDFT give trigonometric interpolation using only a mat-vec $\mathcal{O}(n^2)$.

DFTs

- The DFT y of a real sequence x is generally complex
- Components of y are then conjugate symmetric: $y_k = \bar{y}_{n-k}$, $k = 1, \ldots, (n/2) - 1$
- Two special components:
  1. y_0 is the sum of the components of x. This is the zero frequency component (constant function or DC component)
  2. $y_{n/2}$ has the highest representable frequency (Nyquist frequency)
- Components beyond Nyquist are negatives of those below Nyquist.

Example: DFT

- For a random sequence:

$$y = F_8 x = F_8 \begin{bmatrix} 4 \\ 0 \\ 3 \\ 6 \\ 2 \\ 9 \\ 6 \\ 5 \end{bmatrix} = \begin{bmatrix} 35.0 \\ -5.1 + 8.7i \\ -3.0 + 2.0i \\ 9.1 + 2.7i \\ -5.0 \\ 9.1 - 2.7i \\ -3.0 - 2.0i \\ -5.1 - 8.7i \end{bmatrix}$$

- The transformed sequence is complex, but y_0 and y_4 are real, while y_5, y_6, and y_7 are complex conjugates of y_3, y_2, and y_1, respectively.
- y_0 is in fact the sum

Example: DFT

- For a cyclic sequence:

$$y = F_8 x = F_8 \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 8 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Sequence x has highest rate of oscillation possible for this sample.
- Sequence y has only nonzero component at the Nyquist frequency

Computing the DFT

- Goal: take advantage of symmetries in the DFT for efficiency
- Consider $n = 4$:

$$y_m = \sum_{k=0}^{3} x_k \omega_n^{mk}, \quad m = 0, \ldots, 3$$

or

$$\begin{aligned}
y_0 &= x_0\omega_n^0 + x_1\omega_n^0 + x_2\omega_n^0 + x_3\omega_n^0 \\
y_1 &= x_0\omega_n^0 + x_1\omega_n^1 + x_2\omega_n^2 + x_3\omega_n^3 \\
y_2 &= x_0\omega_n^0 + x_1\omega_n^2 + x_2\omega_n^4 + x_3\omega_n^6 \\
y_3 &= x_0\omega_n^0 + x_1\omega_n^3 + x_2\omega_n^6 + x_3\omega_n^9
\end{aligned}$$

Computing the DFT

- $\omega_n^0 = \omega_n^4 = 1$, $\omega_n^2 = \omega_n^6 = -1$, $\omega_n^9 = \omega_n^1$ so

$$\begin{aligned}
y_0 &= (x_0 + x_2\omega_n^0) + \omega_n^0(x_1 + \omega_n^0 x_3)\\
y_1 &= (x_0 - x_2\omega_n^0) + \omega_n^1(x_1 - \omega_n^0 x_3)\\
y_2 &= (x_0 + x_2\omega_n^0) + \omega_n^2(x_1 + \omega_n^0 x_3)\\
y_3 &= (x_0 - x_2\omega_n^0) + \omega_n^3(x_1 - \omega_n^0 x_3)
\end{aligned}$$

- some of these are 1 (ω_n^0), but even so we have 8 additions and 6 multiplications
- without the simplification this is 12 additions, 16 multiplications

Computing the DFT

- Computing the DFT of the 4 point problem reduced to computing the DFT of its two 2 point even and odd subsequences.
- Generally, the pattern emerges

$$F_1 = 1, \quad F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}, \ldots$$

The DFT of an $n-$point sequence can be computing by two $n/2$-point DFTS (n even)

Computing the DFT

- Let P_4 be a permutation matrix

$$P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and D_2 be the diagonal matrix

$$D_2 = \begin{bmatrix} \omega_4^0 & 0 \\ 0 & \omega_4^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$$

- Then

$$F_4 P_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{bmatrix} = \begin{bmatrix} F_2 & D_2 F_2 \\ F_2 & -D_2 F_2 \end{bmatrix}$$

Computing the DFT

from M. Heath

- So F_4 can be rearranged to diagonally scaled blocks of F_2
- Holds for any even n
- In general, P_n is a permutation that groups even numbered columns then odd numbered columns. And $D_{n/2} = diag(1, \omega_n, \ldots, \omega_n^{(n/2)-1})$.
- To apply F_n, apply $F_{n/2}$ to the even and odd subsequences and scale the results (where necessary) by $\pm D_{n/2}$.
- This recursive DFT is called the Fast Fourier Transform (FFT)

FFT

```
procedure fft($x, y, n, \omega$)
    if $n = 1$ then
        $y[0] = x[0]$                              { bottom of recursion }
    else
        for $k = 0$ to $(n/2) - 1$
            $p[k] = x[2k]$                         { split into even and
            $s[k] = x[2k + 1]$                         odd subsequences }
        end
        fft($p, q, n/2, \omega^2$)                 { call fft procedure
        fft($s, t, n/2, \omega^2$)                     recursively }
        for $k = 0$ to $n - 1$
            $y[k] = q[k \mod (n/2)] +$             { combine results }
                    $\omega^k t[k \mod (n/2)]$
        end
    end
```

FFT

- Cost: $\log_2 n$ levels of recursion, with $\mathcal{O}(n)$ operations each. So a total cost of $\mathcal{O}(n \log_2 n)$.
- Often the transform is computed in-place in one array
- More reductions (storage and operation count in half) if x is real
- Final sequence y can be ordered in $\mathcal{O}(n \log_2 n)$ by a sort
- Can write as iteration rather than recursion.
- Still an algorithm: one particular way of computing the DFT