CS5016: Computational Methods and Applications

Least-Square Function Approximations

Albert Sunny

Department of Computer Science and Engineering
Indian Institute of Technology Palakkad

22 February, 2023

Interpolation vs curve fitting

In function interpolation, our goal was to find a function f that fits some given points $\{(x_i, y_i), i = 1, 2, \ldots, m\}$.

We know that there exists a unique polynomial of degree $m - 1$ that fits m points.

If our goal is to find a cubic polynomial that fits 5 points. This could happen when there are errors/noise. How do we find it?

We know that there is no cubic polynomial that fits all 5 points. Do we consider some 4 out of 5 points? If so, which points do we discard?

We need a way to measure the loss/fitness of a particular curve to a given set of points.

Parameterized functions and a measure of fitness

Let f_θ be a function parameterized by θ; which can be a scalar, vector, finite or countable sequence. E.g.,

$$f_\theta(x) = \sin(\theta x) \qquad \text{or} \qquad f_\theta(\mathbf{x}) = \sum_{i=0}^{k} \theta_i x^i$$

A natural measure of fit is

$$\sum_{i=1}^{m} (y_i - f_\theta(x_i))^2$$

We can then find the best fit curve as follows

$$\min_\theta \sum_{i=1}^{m} (y_i - f_\theta(x_i))^2$$

Best-fit line

$$\min_{a_0, a_1} \sum_{i=1}^{m} [y_i - (a_0 + a_1 x)]^2$$

Show that solving the following equations gives the optimal solution.

Normal equations

$$a_0 m + a_1 \sum_{i=1}^{m} x_i = \sum_{i=1}^{m} y_i$$

$$a_0 \sum_{i=1}^{m} x_i + a_1 \sum_{i=1}^{m} x_i^2 = \sum_{i=1}^{m} x_i y_i$$

Best-fit polynomial

$$\min_{a_0, a_1, \ldots, a_n} \sum_{i=1}^{m} \left[y_i - \left(\sum_{j=0}^{n} a_j x^j \right) \right]^2$$

Show that solving the following equations gives the optimal solution.

Normal equations

$$\sum_{k=0}^{n} a_k \sum_{i=1}^{m} x_i^{j+k} = \sum_{i=1}^{m} y_i x_i^j \quad \forall j \in \{0, 1, \ldots, n\}$$

Least-squares approximation of a function using monomial polynomials

Given a function $f(x)$, continuous on $[a, b]$, find a polynomial $P_n(x)$ of degree at most n

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

such that integral of the square of error is minimized. i.e.,

$$\min_{a_0, a_1, \ldots, a_n} \int_a^b (f(x) - P_n(x))^2 dx$$

We would need to solve the following equations

Normal equations

$$\sum_{k=0}^{n} a_k \cdot \int_a^b x^{j+k} dx = \int_a^b x^j f(x) dx \quad \forall j \in \{0, 1, \ldots, n\}$$

ILL-conditioned matrices!!!

The normal equation has the following matrix form

$$\boldsymbol{S} a = \boldsymbol{b}$$

In the previous methods, matrix $\boldsymbol{S}$ is often **ill-conditioned**.

What is an ill-conditioned matrix? Why do we need to worry about such matrices?

We can make it computationally effective by using special type of polynomials, called **orthogonal polynomials**.

Orthogonal functions

A set of functions $\{\phi_1, \phi_2, \ldots, \phi_n\}$ in $[a, b]$ are called as **orthogonal functions**, with respect to a weight function $w(x)$ if

$$\int_a^b w(x)\phi_i(x)\phi_j(x)dx = \begin{cases} 0 & \text{if } i \neq j \\ c_j & \text{if } i = j \end{cases}$$

where c_j is a positive real number. If $c_j = 1, \forall j$, then the set is called an **orthonormal set**.

Using orthogonal functions

We are interested in finding a least-squares approximation of $f(x)$ on $[a, b]$ by means of a polynomial of the form

$$Q_n(x) = \sum_{i=0}^{n} a_i \phi_i(x)$$

where $\{\phi_i\}_{i=0}^{n}$ is a set of orthogonal polynomials on $[a, b]$, such that the least square error in minimized, i.e.,

$$\min_{a_0, a_1, \ldots, a_n} \int_a^b w(x) \cdot (f(x) - Q_n(x))^2 dx$$

Using orthogonal functions

Setting the partial derivatives to zero, we get

$$\int_a^b w(x)\phi_j(x)f(x)dx = \int_a^b w(x)\phi_j(x)(\sum_{i=0}^{n}\phi_i(x))dx = c_j a_j$$

Or, we have

$$a_j = \frac{1}{c_j}\int_a^b w(x)\phi_j(x)f(x)dx \quad \forall j \in \{0, 1, \ldots, n\}$$

where

$$c_j = \int_a^b w(x)\phi_j^2(x)dx$$

Legendre polynomial[1]

Consider the following polynomials

$$L_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n$$

The above polynomials are orthogonal in the interval $[-1, 1]$ w.r.t. weight function $w(x) = 1$.

What are the first 3 Legendre polynomials? Verify that they are indeed orthogonal.

[1] https://en.wikipedia.org/wiki/Legendre_polynomials

Chebyshev polynomial[2]

Consider the following polynomial

$$T_n(x) = \cos(n \cos^{-1}(x))$$

Is $T_n(x)$ really a polynomial? In fact, we have

$$T_0(x) = 1 \qquad \text{and} \qquad T_1(x) = x$$

Further, we have the following recurrence relation

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Try to prove the above recurrence relations.

Chebyshev polynomials are orthogonal in the interval $[-1, 1]$ w.r.t. weight function $w(x) = 1/\sqrt{1-x^2}$.

[2] https://en.wikipedia.org/wiki/Chebyshev_polynomials

Fourier Series

For any positive integer n, the set of functions $\{\cos(0), \cos(x), \ldots \cos(nx), \sin(0), \sin(x), \ldots \sin(nx)\}$ is **orthogonal** in the interval $[-\pi, \pi]$ with respect to the weight function $w(x) = 1$.

Try to verify the above statement.

Let

$$S_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n} a_k \cos(kx) + \sum_{k=1}^{n} b_k \sin(kx)$$

such that the least square error in minimized, i.e.,

$$\min \int_{-\pi}^{\pi} (f(x) - S_n(x))^2 dx$$

Equating partial derivatives to zero, due to orthogonality, we get

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx, \quad b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx$$

Discrete Fourier Transform

Suppose we have $2m$ data points x_k, y_k where

$$x_k = -\pi + \frac{k\pi}{m} \text{ and } y_k = f(x_k),\ k \in \{0, 1, \ldots, 2m-1\}$$

The discrete least squares fit of a trigonometric polynomial does the following

$$\min \sum_{k=0}^{2m-1} (S_n(x_k) - y_k)^2$$

Discrete Fourier Transform

Lemma

If r is not a multiple of $2m$,

$$\sum_{k=0}^{2m-1} \cos(rx_k) = \sum_{k=0}^{2m-1} \sin(rx_k) = 0$$

Lemma

If $r \neq 0$ is not a multiple of m,

$$\sum_{k=0}^{2m-1} [\cos(rx_k)]^2 = \sum_{k=0}^{2m-1} [\sin(rx_k)]^2 = m$$

Discrete Fourier Transform

Lemma

If $r \neq l$ and $r + l$ is not a multiple of $2m$,

$$\sum_{k=0}^{2m-1} \cos(rx_k)\cos(lx_k) = \sum_{k=0}^{2m-1} \sin(rx_k)\sin(lx_k) = 0$$

$$\sum_{k=0}^{2m-1} \cos(rx_k)\sin(lx_k) = \sum_{k=0}^{2m-1} \sin(rx_k)\cos(lx_k) = 0$$

Discrete Fourier Transform

Then, for any $n < m$, the best approximation is

$$S_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n} a_k \cos(kx) + \sum_{k=1}^{n} b_k \sin(kx)$$

Due the previous 3 lemmas, we have

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cdot cos(kx_j) \quad \text{and} \quad b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cdot sin(kx_j)$$

Let us choose $n = m - 1$. Then,

$$\{y_j\}_{j=0}^{2m-1} \xrightarrow{DFT} \{(a_k, b_k)\}_{k=0}^{m-1}$$

Is there any issue if $n \geq m$?

Fast Fourier Transform (FFT)

We need $O(m^2)$ operations to compute $\{(a_k, b_k)\}_{k=0}^{m-1}$.

However, there is a fast $O(m \log_2(m))$ algorithm known as *Fast Fourier Transform*[3] that can compute these coefficients.

The SciPy module `scipy.fft` is a more comprehensive package for discrete Fourier transform. To know more visit `https://docs.scipy.org/doc/scipy/reference/fft.html`

To know about a Python sub-package for efficiently dealing with polynomials, visit `https://numpy.org/doc/stable/reference/routines.polynomials.package.html`

[3] `https://en.wikipedia.org/wiki/Fast_Fourier_transform`

Thank You