Interpolation and extrapolation

Alexander Khanov

PHYS6260: Experimental Methods is HEP
Oklahoma State University

October 28, 2013
Interpolation/extrapolation vs fitting

- Formulation of the problem:
  - there are two quantities \( x \) and \( y \) related by some (generally unknown) functional dependence \( y = \varphi(x) \)
  - we have a set of \( N \) independent measurements of \( y \): \( y = y_1, \ldots, y_N \)
    taken at \( N \) values of \( x \): \( x = x_1, \ldots, x_N \)
  - we want to evaluate \( y \) at arbitrary \( x \)

- Data fit:
  - \( y = y_1, \ldots, y_N \) have known variances \( \sigma_1^2, \ldots, \sigma_N^2 \)
  - we want to find a function \( f(x, p) \) which depends on \( n < N \) parameters \( p = p_1, \ldots, p_n \) where \( p \) are chosen in some optimal way, e.g. minimize the overall difference between \( y_i \) and \( f(x_i) \)
  - methods: least squares, maximum likelihood

- Data interpolation/extrapolation:
  - we want to find a function \( f(x) \) such that for all \( i = 1, \ldots, N \):
    \( f(x_i) = y_i \)
  - the function must be “smooth” and not far away from \( \varphi(x) \)
Linear interpolation

- For each $i$, define $F_i(x) = \frac{(x - x_i)y_{i+1} + (x_{i+1} - x)y_i}{x_{i+1} - x_i}$
  - $F_i(x)$ is linear
  - $F_i(x_i) = y_i$, $F_i(x_{i+1}) = y_{i+1}$
- Simple, easy to calculate, well behaved
- No good if need to estimate derivatives
Lagrange interpolation

Since all “good” functions can be expanded in a Taylor series, it makes sense to take \( f(x) \) as a polynomial unless it’s a degenerate case, having \( N \) coefficients is necessary and sufficient to satisfy \( N \) conditions \( f(x_i) = y_i \)

How to construct such a polynomial?

\[
(x - x_1) \ldots (x - x_{k-1})(x - x_{k+1}) \ldots (x - x_N)
\]
is a polynomial of degree \( N - 1 \) which is equal to zero at all \( x = x_i \neq x_k \)

\[
F_i(x) = \frac{(x - x_1) \ldots (x - x_{k-1})(x - x_{k+1}) \ldots (x - x_N)}{(x_k - x_1) \ldots (x_k - x_{k-1})(x_k - x_{k+1}) \ldots (x_k - x_N)}
\]
is a polynomial of degree \( N - 1 \) which is equal to zero at all \( x = x_i \neq x_k \) and equal to one at \( x = x_k \)

\[
f(x) = \sum_{i=1}^{N} y_i F_i(x)
\]
is a polynomial of degree \( N - 1 \) which satisfies \( f(x_i) = y_i \) for all \( i \)
Lagrange interpolation (2)
How good is polynomial interpolation?

- Since we don’t know $\varphi(x)$ we can’t say for certain, but in general for any $x$ there is $\xi$ such that

\[ \varphi(x) - f(x) = \frac{\varphi^{(N)}(\xi)}{N!} \prod_{i=1}^{N} (x - x_i) \]

- If we want to optimize the interpolation by adjusting the points $x_i$ where we take the measurements, then we should try to minimize the maximum $\prod_{i=1}^{N} (x - x_i)$ (so called minimax principle)
Problems of polynomial interpolation

- Depending on $\varphi(x)$, the polynomial interpolation can exhibit oscillating behavior as the number of points increases ("Runge's phenomenon")

\[ \varphi = \frac{1}{1 + (5x - 10)^2} \]
Chebyshev polynomials

- If the choice of points $x_i$ is up to us, one can use Chebyshev polynomials
  \[ T_n(x) = \cos(n \cos^{-1} x) \quad -1 \leq x \leq 1 \]
  \[ T_1(x) = 1 \quad T_2(x) = x \quad T_3(x) = 2x^2 - 1 \quad T_4(x) = 4x^3 - 3x \quad \ldots \]

  \( T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \) and tons of other funny properties

- The trick is to take the roots of $N$th Chebyshev polynomial as the measurement points:
  \[ x_i = \cos \left( \pi \frac{2i - 1}{2N} \right), \quad i = 1, \ldots, N \]

  One can show that this choice of $x_i$ is optimal in the minimax sense
Cubic splines

- The polynomial interpolation provides a “single formula” solution with nice mathematical properties but it’s often an overkill
  - a simpler solution: a piecewise low degree polynomial which provides a smooth curve
- Cubic spline: for each interval $x_i < x < x_{i+1}$, define a cubic polynomial $F_i(x) = a_0 + a_1x + a_2x^2 + a_3x^3$
  - four unknowns $\Rightarrow$ need four equations
- Two conditions per point come from $F_i(x_i) = y_i$, $F_i(x_{i+1}) = y_{i+1}$
- Two more conditions per point make sure that first and second derivatives are continuous at join points
- One also needs two more conditions at the end points
  - e.g. “natural spline” $F_1''(x_1) = 0$, $F_N''(x_N) = 0$
- Given the $x_i$, $y_i$, the spline coefficients are found from a system of linear equations
- It’s also possible to do a combination of splines and fitting when the spline join points (“knots”) are chosen different from the actual measurement points
Extrapolation

- Extrapolation: want to know $\varphi(x)$ outside the data range (for $x < x_1$ or $x > x_N$)
- Can use polynomial extrapolation, but no error estimate is available!
Data smoothing

"Fitting" data without using any explicit fitting function

The idea is to get rid of noise while preserving the important pattern properties

Usual smoothing techniques:

▶ running medians, e.g. \( z_i = \text{median}(y_{i-1}, y_i, y_{i+1}) \)

▶ running means, e.g. \( z_i = \frac{(y_{i-1} + y_i + y_{i+1})}{3}, \)
\[ z_i = \frac{y_{i-1}}{4} + \frac{y_i}{2} + \frac{y_{i+1}}{4} \]

▶ popular algorithm (back to 70's): 353QH

Usually smoothing is neither desirable nor permissible