Local Regression Methods in PyDelt

Michael Lee

July 14, 2025

Abstract

This document provides a detailed explanation of the local regression methods implemented in the PyDelt library, with a focus on their relationship to LOESS (LOcally Estimated Scatterplot Smoothing) and other local regression techniques. We discuss the theoretical foundations, implementation details, and comparative advantages of each method for derivative estimation and raw interpolation.

1 Introduction

Local regression methods are non-parametric techniques that fit simple models to localized subsets of data to build a function that describes the deterministic part of the variation in the data. The PyDelt library implements several local regression methods for derivative estimation, including:

- Local Linear Approximation (LLA)
- Generalized Local Linear Approximation (GLLA)
- Generalized Orthogonal Local Derivative (GOLD)
- Functional Data Analysis (FDA) using splines

This document explores how these methods relate to traditional LOESS and other local regression techniques used for raw interpolation.

2 LOESS and Local Regression

2.1 LOESS Overview

LOESS (LOcally Estimated Scatterplot Smoothing), also known as LOWESS (LOcally WEighted Scatterplot Smoothing), is a popular non-parametric regression method that combines multiple regression models in a k-nearest-neighbor-based meta-model. The key characteristics of LOESS include:

- Fitting simple models (typically low-degree polynomials) to localized subsets of the data

- Using weighted least squares with a weight function that gives more weight to points near the point of estimation and less weight to points further away
- Robust fitting procedures to reduce the influence of outliers

The general form of LOESS involves fitting a polynomial regression of degree d (typically 1 or 2) to a subset of the data using weighted least squares, where the weights decrease with distance from the point of estimation.

2.2 Mathematical Formulation

For a dataset (x_i, y_i) for $i = 1, 2, \ldots, n$, LOESS estimates the value at a point x by:

1. Selecting k nearest neighbors to x
2. Assigning weights w_i to each neighbor based on their distance from x, typically using a tri-cubic weight function:

   $$w_i = \left(1 - \left|\frac{x - x_i}{h}\right|^3\right)^3 \tag{1}$$

   where h is the maximum distance from x to the k-th nearest neighbor
3. Fitting a weighted polynomial regression of degree d to the k nearest neighbors
4. Using the fitted polynomial to estimate the value at x

3 Local Linear Approximation (LLA)

3.1 Method Description

The Local Linear Approximation (LLA) method in PyDelt estimates derivatives by fitting linear models to localized subsets of the time series data. It uses a sliding window approach and applies optional normalization techniques.

3.2 Relation to LOESS

LLA can be viewed as a simplified version of LOESS with the following characteristics:

- Uses a fixed-width window rather than k-nearest neighbors
- Employs a polynomial of degree 1 (linear regression)
- Does not use distance-based weighting within the window (uniform weights)

- Offers options for zero-min and zero-mean normalization within each window

The key difference is that LLA focuses on estimating the slope (first derivative) rather than the function value itself. The slope of the fitted line directly provides the derivative estimate.

3.3 Mathematical Formulation

For a time series (t_i, s_i) for $i = 1, 2, \ldots, n$, LLA estimates the derivative at a point t_j by:

1. Selecting a window of size w centered at t_j
2. Optionally normalizing the data within the window by:
   - Zero-min: $t'_i = t_i - \min(t_i)$, $s'_i = s_i - \min(s_i)$
   - Zero-mean: $t'_i = t_i - \text{mean}(t_i)$, $s'_i = s_i - \text{mean}(s_i)$
3. Fitting a linear regression $s'_i = \beta_0 + \beta_1 t'_i + \epsilon_i$ to the data in the window
4. Using the slope coefficient β_1 as the derivative estimate at t_j

4 Generalized Local Linear Approximation (GLLA)

4.1 Method Description

The Generalized Local Linear Approximation (GLLA) extends LLA to estimate higher-order derivatives using a Taylor series expansion approach. It fits a polynomial of specified degree to a sliding window of data points.

4.2 Relation to LOESS

GLLA relates to LOESS in the following ways:

- Uses a fixed-width window similar to LLA
- Fits higher-degree polynomials (like LOESS with $d > 1$)
- Does not use distance-based weighting
- Focuses on extracting derivative information rather than function values

The key innovation in GLLA is the use of the Taylor series coefficients to directly estimate derivatives of various orders.

4.3 Mathematical Formulation

For a time series (t_i, s_i) for $i = 1, 2, \ldots, n$, GLLA estimates derivatives up to order m at a point t_j by:

1. Creating a design matrix L where each column represents a term in the Taylor series:
$$L_{ij} = \frac{(t_i - \bar{t})^j}{j!} \tag{2}$$
for $j = 0, 1, \ldots, m$

2. Computing the weight matrix:
$$W = L(L^T L)^{-1} \tag{3}$$

3. Applying the weights to the signal values to estimate derivatives:
$$\hat{s}^{(j)} = XW_j \cdot \Delta t^{-j} \tag{4}$$
where X is the matrix of signal values in the sliding window and Δt is the time step

5 Generalized Orthogonal Local Derivative (GOLD)

5.1 Method Description

The Generalized Orthogonal Local Derivative (GOLD) method improves upon GLLA by using orthogonalized basis functions. It applies Gram-Schmidt orthogonalization to the polynomial basis functions before estimating derivatives.

5.2 Relation to LOESS

GOLD relates to LOESS and local regression in the following ways:

- Uses a fixed-width window like GLLA and LLA
- Employs orthogonalized polynomial basis functions, which improves numerical stability
- Does not use distance-based weighting
- Focuses on derivative estimation rather than function values

The orthogonalization step in GOLD is similar to techniques used in orthogonal polynomial regression, which can be viewed as a specialized form of local regression.

5.3 Mathematical Formulation

For a time series (t_i, s_i) for $i = 1, 2, \ldots, n$, GOLD estimates derivatives up to order m at a point t_j by:

1. Creating a basis of polynomial functions:

$$\phi_k(t) = t^k \quad \text{for } k = 0, 1, \ldots, m \tag{5}$$

2. Orthogonalizing the basis using Gram-Schmidt:

$$\phi'_k(t) = \phi_k(t) - \sum_{j=0}^{k-1} \frac{\langle \phi_k, \phi'_j \rangle}{\langle \phi'_j, \phi'_j \rangle} \phi'_j(t) \tag{6}$$

3. Scaling the orthogonalized basis by factorial terms:

$$L_k = \frac{1}{k!} \phi'_k \tag{7}$$

4. Computing the weight matrix:

$$W = L^T (LL^T)^{-1} \tag{8}$$

5. Applying the weights to the signal values to estimate derivatives

6 Functional Data Analysis (FDA)

6.1 Method Description

The Functional Data Analysis (FDA) method in PyDelt uses spline smoothing to represent the time series as a continuous function. It then analytically computes derivatives from the spline representation.

6.2 Relation to LOESS

FDA differs from LOESS and the other local regression methods in several ways:

- Uses a global spline representation rather than local polynomials
- Employs a smoothing parameter to control the trade-off between fit and smoothness
- Computes derivatives analytically from the spline rather than from local fits
- Provides a continuous representation of the function and its derivatives

While not a local regression method in the strict sense, FDA shares the goal of providing a smooth representation of the data from which derivatives can be estimated.

6.3 Mathematical Formulation

For a time series (t_i, s_i) for $i = 1, 2, \ldots, n$, FDA estimates the function and its derivatives by:

1. Fitting a smoothing spline $f(t)$ that minimizes:

$$\sum_{i=1}^{n}(s_i - f(t_i))^2 + \lambda \int f''(t)^2 dt \qquad (9)$$

   where λ is the smoothing parameter

2. Computing derivatives analytically from the spline representation:

$$f^{(k)}(t) = \frac{d^k}{dt^k} f(t) \qquad (10)$$

7 Comparative Analysis

7.1 Advantages and Limitations

Feature	**LLA**	**GLLA**	**GOLD**	**FDA**
Computational Complexity	Low	Medium	High	Medium
Higher-order Derivatives	No	Yes	Yes	Yes
Noise Sensitivity	Medium	Medium	Low	Low
Parameter Tuning	Simple	Medium	Medium	Complex
Boundary Handling	Poor	Poor	Poor	Good

Table 1: Comparison of local regression methods in PyDelt

7.2 Use Cases

- **LLA**: Best for simple first-derivative estimation with minimal computational overhead
- **GLLA**: Suitable for higher-order derivatives in well-behaved data
- **GOLD**: Preferred for higher-order derivatives in noisy data due to improved numerical stability
- **FDA**: Ideal for smooth data where a continuous representation is desired

8 Conclusion

The local regression methods implemented in PyDelt provide a range of approaches for derivative estimation, each with its own strengths and limitations. While they share conceptual similarities with LOESS and other local regression techniques used for raw interpolation, they are specifically designed for derivative estimation.

The key differences between these methods and traditional LOESS include:

- Focus on derivative estimation rather than function value prediction
- Different approaches to window selection and weighting
- Specialized techniques for higher-order derivative estimation

Understanding these relationships helps users select the appropriate method for their specific application and interpret the results correctly.