

Locally Weighted Regression

Aayush Bajaj

July 10, 2026

Contents

1	motivation	1
2	the estimator	1
2.1	weighted least squares at a query point	1
2.2	kernels	2
3	bandwidth is the bias-variance dial	2
4	connection to kernel smoothing	3
5	in code	4
6	see also	5

a straight line is too rigid for a wiggly world, and a global degree-9 polynomial is a hostage negotiation. locally weighted regression (LWR — and its robust cousin LOWESS) takes the diplomatic route: fit the simplest possible model, but fit it *freshly at every query point*, paying attention only to the training points nearby.

the third option — splines — is a different page's problem.

1 motivation

- linear regression commits to one θ for the whole input space. if the true f bends, the residue of that commitment is bias everywhere.
- the fix need not be a fancier global family. **any smooth function is locally linear** — taylor says so — so a linear fit weighted toward a neighbourhood of x_0 can track an arbitrary smooth f .
- the price: there is no longer a “trained model”. LWR is **memory-based and non-parametric** — like **k-nearest neighbours**, it keeps the entire training set and does all of its work at prediction time. training is $O(1)$; every query costs a fresh weighted least-squares solve.

the classical name for this family is ‘lazy learning’, which is also a fair description of the author.

2 the estimator

2.1 weighted least squares at a query point

fix a query x_0 . assign each training point a weight $w_i(x_0) \geq 0$ that decays with distance from x_0 , then solve the weighted least-squares problem

$$\hat{\theta}(x_0) = \arg \min_{\theta} \sum_{i=1}^n w_i(x_0) (y_i - \theta^\top x_i)^2. \quad (1)$$

stack the weights into $W = \text{diag}(w_1(x_0), \dots, w_n(x_0))$. setting the gradient $-2X^\top W(y - X\theta)$ to zero gives the closed form

$$\boxed{\hat{\theta}(x_0) = (X^\top W X)^{-1} X^\top W y} \quad (2)$$

and the prediction is $\hat{f}(x_0) = x_0^\top \hat{\theta}(x_0)$. ordinary least squares is the special case $W = I$ — LWR is just **OLS** wearing distance-tinted glasses.

- in practice one augments x with a constant (and, for *local polynomial* regression, powers of $x - x_0$): degree 0 is a weighted average, degree 1 a weighted line, degree 2 a weighted parabola.
- note $\hat{\theta}$ depends on x_0 through W : a different regression for every query. the final curve $x_0 \mapsto \hat{f}(x_0)$ is smooth even though no single global model produced it.
- each prediction is **linear in** y : $\hat{f}(x_0) = \ell(x_0)^\top y$ for a weight vector $\ell(x_0)$ that depends only on the inputs. LWR is a *linear smoother*, so an effective degrees of freedom exists: $\text{df} = \text{tr}(L)$, where row i of L is $\ell(x_i)^\top$ (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

2.2 kernels

the weight function is a kernel $w_i(x_0) = K_\lambda(x_0, x_i)$. two standard choices:

- **gaussian**: $K(x_0, x_i) = \exp(-\|x_i - x_0\|^2 / 2\tau^2)$. every point gets some weight; τ is the **bandwidth**.
- **tricube** (the LOWESS default): with $u = |x_i - x_0|/\lambda$,

$$K(u) = \begin{cases} (1 - u^3)^3 & u < 1 \\ 0 & u \geq 1, \end{cases} \quad (3)$$

compactly supported (points beyond λ contribute exactly nothing — cheaper, and the fit at x_0 is immune to far-away outliers) and smooth at the boundary of its support (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).

- LOWESS makes the bandwidth **adaptive**: $\lambda(x_0)$ is set to the distance to the k -th nearest neighbour, so a fixed *fraction* (the “span”) of the data is in play everywhere — wide windows where data are sparse, narrow where dense.¹

3 bandwidth is the bias-variance dial

everything you know from the **bias-variance decomposition** applies with τ as the knob:

¹ cleveland (1979), *robust locally weighted regression and smoothing scatterplots*, jasa 74(368). the “robust” part: after fitting, LOWESS downweights points with large residuals (bisquare weights) and refits — a few iterations of built-in outlier resistance.

bandwidth	behaviour
$\tau \rightarrow 0$	only the nearest point matters: interpolation, zero bias, maximal variance
τ small	wiggly curve chasing noise — variance dominates
τ large	window covers everything: collapses to global OLS — bias dominates
$\tau \rightarrow \infty$	exactly global least squares

the sweet spot minimises estimated test error, and since each fit is cheap and the model never “trains”, leave-one-out **cross validation** is the natural selector — for linear smoothers it even has a closed form, no refitting required.

4 connection to kernel smoothing

LWR is the polished member of a family (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009):

- **degree 0** is the **nadaraya–watson kernel smoother**: minimising $\sum_i w_i (y_i - \theta_0)^2$ over a constant θ_0 gives the kernel-weighted average

$$\hat{f}(x_0) = \frac{\sum_i K_\lambda(x_0, x_i) y_i}{\sum_i K_\lambda(x_0, x_i)}. \quad (4)$$

- but a locally-constant fit has a defect at the edges: near a boundary the window is one-sided, so if f is sloping, the average of the neighbours sits systematically above or below the truth. this is **boundary bias**, and it is first-order — it does not vanish quickly with more data.
- **local linear regression fixes it automatically**: fitting a slope lets the estimator extrapolate the trend across the asymmetric window. local linear fits are exact for linear f everywhere, boundaries included, killing the first-order term in the bias expansion (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009). the code below makes this concrete — watch the two estimators disagree at $x_0 = 0$ and $x_0 = 1$.
- local *quadratic* fits go one better in curved interior regions (locally-linear fits trim the hills and fill the valleys — curvature bias) at the price of higher variance. degree 1 is the everyday compromise.
- the general kernel view of learning — feature spaces, gram matrices, representer theorems — lives on the **kernel methods** page; here the kernel is doing a humbler job, defining “nearby”.

one warning from the **curse of dimensionality**: local methods lean entirely on neighbourhoods being small and populated. in high dimensions neighbourhoods that contain enough points stop being local, boundary points become the majority, and LWR quietly degrades. it is a low-dimensional specialist.

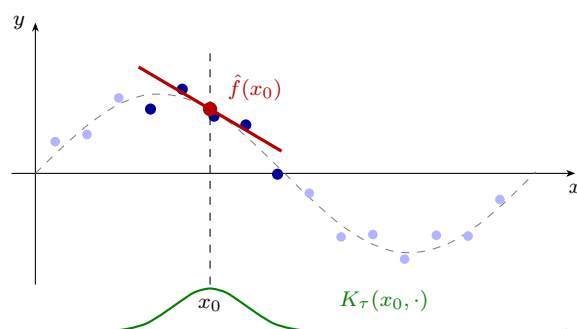


Figure 1: one query, one fit: the kernel (bottom) weights the training points near x_0 ; a weighted line is fitted (solid segment) and read off at x_0 only. slide x_0 across the axis and the evaluations trace out a smooth curve.

5 in code

LWR from scratch on a noisy sine — gaussian kernel, local polynomials of degree 0 (nadaraya–watson) and degree 1, and a bandwidth sweep:

```

1  import numpy as np
2
3  rng = np.random.default_rng(7)
4  f = lambda x: np.sin(2 * np.pi * x)
5  n, sigma = 60, 0.25
6  x = np.sort(rng.uniform(0, 1, n))
7  y = f(x) + rng.normal(0, sigma, n)
8
9  def lwr(x0, x, y, tau, degree=1):
10     """fit a weighted polynomial at query x0, return prediction there."""
11     w = np.exp(-((x - x0) ** 2) / (2 * tau ** 2)) # gaussian kernel
12     A = np.vander(x - x0, degree + 1, increasing=True) # [1, (x-x0), ...]
13     W = np.diag(w)
14     theta = np.linalg.solve(A.T @ W @ A, A.T @ W @ y) # (A'WA)^{-1} A'Wy
15     return theta[0] # value at x0
16
17  xg = np.linspace(0, 1, 101)
18  print(f"{'tau':>6} {'rmse (deg 0)':>13} {'rmse (deg 1)':>13}")
19  for tau in [0.02, 0.05, 0.1, 0.2, 0.5, 2.0]:
20     for deg in (0, 1):
21         yhat = np.array([lwr(x0, x, y, tau, deg) for x0 in xg])
22         rmse = np.sqrt(((yhat - f(xg)) ** 2).mean())
23         if deg == 0:
24             row = f"{'tau':>6.2f} {'rmse:>13.4f}"
25         else:
26             row += f" {'rmse:>13.4f}"
27     print(row)
28
29  # a few point predictions at the best bandwidth, incl. the boundary
30  tau = 0.1
31  print("\n x0      truth    deg-0    deg-1")
32  for x0 in [0.00, 0.25, 0.50, 0.75, 1.00]:
33     p0, p1 = lwr(x0, x, y, tau, 0), lwr(x0, x, y, tau, 1)
34     print(f"{'x0':>7.2f} {'f(x0):>8.3f} {'p0:>7.3f} {'p1:>7.3f}")

```

tau	rmse (deg 0)	rmse (deg 1)
0.02	0.1327	0.1268
0.05	0.0773	0.0674
0.10	0.1620	0.1231
0.20	0.3455	0.2812
0.50	0.5892	0.4374
2.00	0.6956	0.4523

x0	truth	deg-0	deg-1
0.00	0.000	0.253	-0.052
0.25	1.000	0.843	0.843
0.50	0.000	-0.052	-0.022
0.75	-1.000	-0.808	-0.806
1.00	-0.000	-0.459	-0.043

three lessons in one table:

- **bandwidth u-shape.** rmse bottoms out near $\tau = 0.05$ and climbs in both directions: at $\tau = 0.02$ the curve chases individual noisy points (variance), by $\tau = 0.5$ the “local” window spans half a period of the sine (bias), and $\tau = 2$ is effectively a global fit to a sine wave.
- **boundary bias, live.** at the interior points the two degrees agree closely. at $x_0 = 0$ and $x_0 = 1$ the degree-0 smoother misses by 0.25 and 0.46 — its one-sided window averages points that all sit above (respectively below) the truth. degree 1 extrapolates the local trend and lands within 0.06. exactly the theory’s prediction.
- **interior peaks are trimmed.** both estimators read 0.84 at the crest $x_0 = 0.25$ where the truth is 1 — smoothing flattens local maxima (curvature bias); a local quadratic would recover most of it.

complexity check: every query re-solves a small $(d+1) \times (d+1)$ system built from all n points — $O(n)$ work per prediction, versus $O(1)$ for a fitted parametric model. with a compact kernel and a k-d tree the constant improves, but the memory-based character remains.

6 see also

- **ordinary least squares** — the $W = I$ special case, and the engine inside every local fit
- **k-nearest neighbours** — the same memory-based bet with a cruder (rectangular) kernel
- **bias-variance decomposition** — what the bandwidth is really adjusting
- **cross validation** — choosing τ , with a free closed form for linear smoothers
- **curse of dimensionality** — why this page lives happily in 1–3 dimensions and dies in 30

references

Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer.