Yale University

Nonlinear Regression and the Bias–Variance Tradeoff

S&DS 265 · Introductory Machine Learning · Lecture 6

Zhuoran Yang

Fall 2026

AI usage: figures were generated, and these slides polished, with the help of AI tools.

Outline

Learning Objectives

In this lecture you will learn:

1. The setup of general regression task;
2. How parametric and nonparametric model classes differ;
3. How polynomial, kernel, and fully connected network regressors are defined;
4. Why local fitting becomes difficult as the number of features grows;
5. How bias, variance, validation, and extrapolation guide model selection.

Outline

Motivating Example: Wage and Age

A labor economist has wage records for 3,000 workers and wants to predict annual wage for another worker from age.

A linear model assumes the same change per year at ages 25, 45, and 65.

What model can represent a relationship that rises, levels off, or bends while still predicting an unseen worker?

Example and data: ISLP §7.1 and Lab 7 (James, Witten, Hastie, Tibshirani, and Taylor, 2023).

The Wage Data

worker	year	age	education	wage ($1000s)
1	2006	18	< high school	75.04
2	2004	24	college	70.48
3	2003	45	some college	130.98
4	2003	43	college	154.69
⋮	⋮	⋮	⋮	⋮

One row is one male worker from the Mid-Atlantic region. We begin with age as the only input and annual wage as the response.

Source: The `Wage` data, ISLP §7.1; available at `statlearning.com`.

Reading One Row

worker	year	age	education	wage ($1000s)
1	2006	18	< high school	75.04

Worker 1 was 18 years old in 2006, had not completed high school, and earned about $75,043 annually. In today's one-input problem, $x_1 = 18$ and $y_1 = 75.04$; the other columns are observed but not used.

Problem Setup

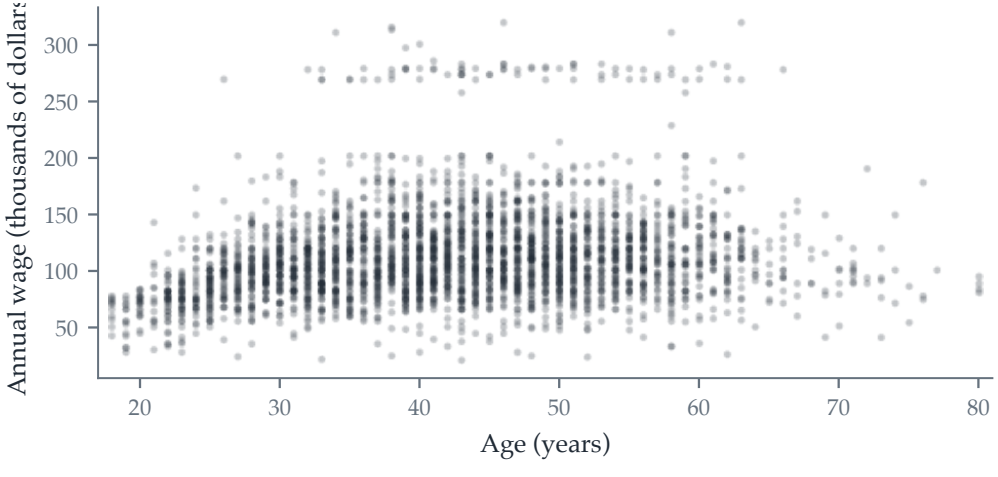
Training data. $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^{3000}$ contains worker records, treated as independently sampled for this analysis.

Input. $x_i \in \mathbb{R}$ is age in years.

Response. $y_i \in \mathbb{R}$ is annual wage in thousands of dollars.

Goal. Predict wage for an unseen worker whose age lies within the observed range, 18–80 years.

Observed Wage and Age



The `Wage` data, ISLP §7.1; each point is one observed worker.

Data Assumption and Regression Target

As in Lecture 3, suppose

$$Y = m(X) + \varepsilon, \qquad \mathbb{E}[\varepsilon \mid X] = 0, \qquad \mathrm{Var}(\varepsilon \mid X = x) = \sigma^2(x).$$

Taking the conditional expectation at a fixed age gives

$$\mathbb{E}[Y \mid X = x] = m(x) + \mathbb{E}[\varepsilon \mid X = x] = \boxed{m(x)}.$$

Recall that using the squared-loss function, the regression target is $m(x) = \mathbb{E}[Y \mid X = x]$ — $m(x)$ minimizes the square-loss over all functions.

Three Choices in Nonlinear Regression

1. **Model class:** choose a set $\mathcal{F}$ of candidate functions that can bend with x.
2. **Objective:** compare candidates by empirical squared loss,
$$\widehat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2.$$
3. **Solution:** compute $\hat{f} \in \arg\min_{f\in\mathcal{F}} \widehat{R}_n(f)$ using the structure of the class.

Different choices of $\mathcal{F}$ produce different optimization problems.

Candidate model classes: linear models, polynomials, kernel estimators, and neural networks.

Parametric and Nonparametric Classes

Parametric

$$\mathcal{F} = \{f_\theta : \theta \in \Theta \subseteq \mathbb{R}^p\}.$$

Each element in $\mathcal{F}$ has a parameter θ. The parameter dimension p is fixed when the class is chosen.

Examples: degree-r polynomials; a fully connected network with fixed depth and width.

Nonparametric

$\mathcal{F}_n$ may grow richer as n grows.

Local resolution and effective complexity depend on the data and a smoothing parameter.

Examples: kernel regression; nearest-neighbor regression; smoothing splines.

Training Error under Nested Classes

If model classes are nested, $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots$, then

$$\min_{f \in \mathcal{F}_{r+1}} \widehat{R}_n(f) \leq \min_{f \in \mathcal{F}_r} \widehat{R}_n(f).$$

The larger class can always reproduce the smaller class's fitted function, so training error cannot rise. Test error has no such guarantee because the selected function $\arg\min_{f \in \mathcal{F}_r} \widehat{R}_n(f)$ is data-dependent and thus also responds to sample-specific noise.

Outline

Basis Expansions

Choose functions $\phi_0, \dots, \phi_r$ before fitting and define

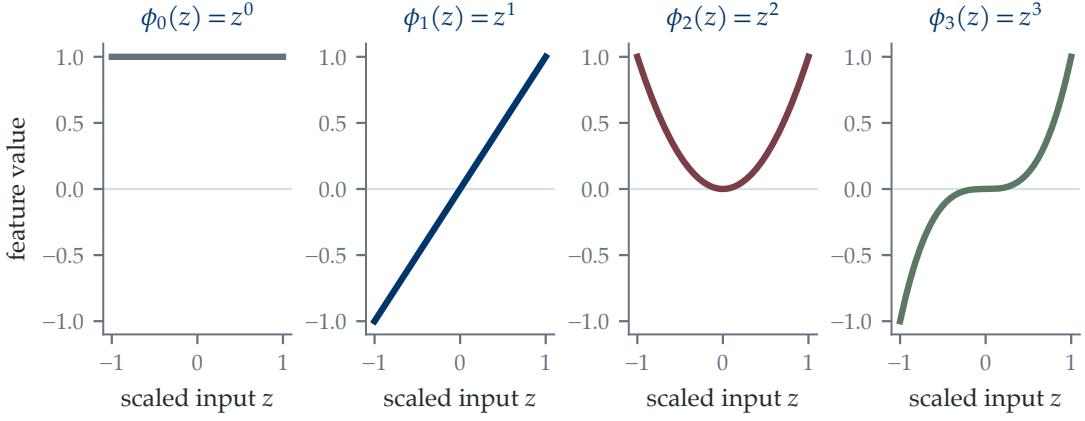
$$\phi(x) = (\phi_0(x), \dots, \phi_r(x))^\top, \qquad f_\beta(x) = \beta^\top \phi(x).$$

The model can be nonlinear in the raw input x while remaining linear in the fitted coefficients β. That is, $\phi(x)$ is nonlinear in x but f_θ is linear in $\phi(x)$.

For observations $x_1, \dots, x_n$, the design matrix is

$$\Phi_{ij} = \phi_j(x_i), \qquad \Phi \in \mathbb{R}^{n \times (r+1)}.$$

Polynomial Basis Functions



Course illustration on a scaled input $z \in [-1, 1]$.

Polynomial Function Classes

The degree-r polynomial class is

$$\mathcal{F}_r^{\text{poly}} = \left\{ f_\beta(x) = \sum_{j=0}^{r} \beta_j x^j : \beta \in \mathbb{R}^{r+1} \right\}.$$

The classes are nested:

$$\mathcal{F}_1^{\text{poly}} \subset \mathcal{F}_2^{\text{poly}} \subset \cdots,$$

because a degree-r polynomial is also degree $r+1$ with $\beta_{r+1} = 0$. Degree is a model-class choice, and should be set a priori.

Polynomial Least Squares

derive

Fitting over $\mathcal{F}_r^{\text{poly}}$ is ordinary least squares on the constructed features:

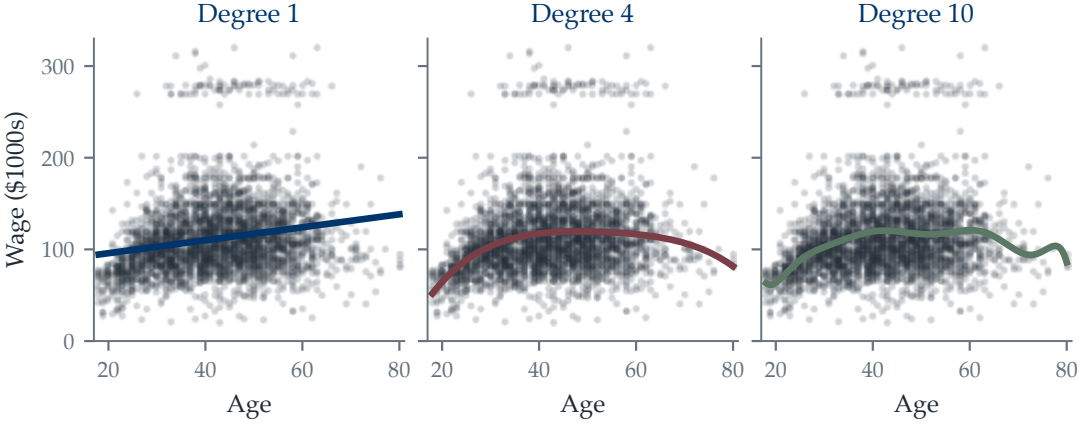
$$\hat{\beta}_r = \arg\min_{\beta \in \mathbb{R}^{r+1}} \frac{1}{n} \|y - \Phi_r \beta\|_2^2.$$

Differentiating gives

$$-\frac{2}{n} \Phi_r^\top (y - \Phi_r \beta) = 0 \quad \Longrightarrow \quad \Phi_r^\top \Phi_r \hat{\beta}_r = \Phi_r^\top y.$$

Thus the solution machinery from Lectures 3 and 5 applies unchanged.

Polynomial Regression on Wage



Course least-squares fits to the ISLP `Wage` data; dark points are observed workers.

A Depth-One Fully Connected Network

Choose the width H a priori. For $j = 1, \dots, H$, define

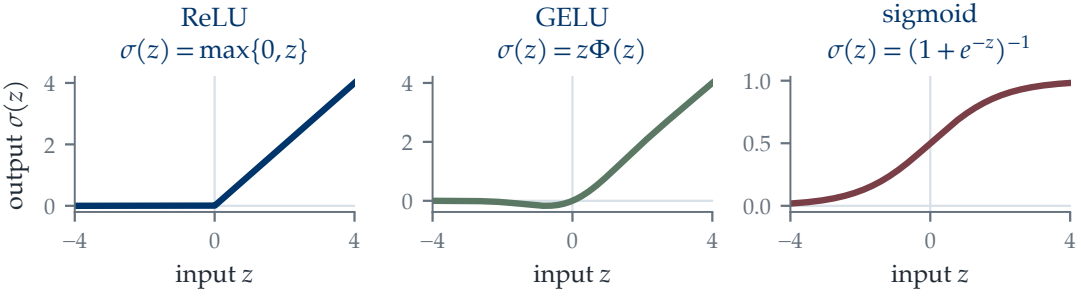
$$h_j(x) = \sigma(w_j^\top x + b_j), \qquad \boxed{f_\theta(x) = \sum_{j=1}^{H} a_j h_j(x)}.$$

$$\underbrace{x \in \mathbb{R}^d}_{\text{observed input}} \longrightarrow \underbrace{h(x) = (h_1(x), \dots, h_H(x))^\top \in \mathbb{R}^H}_{\text{hidden layer}} \longrightarrow \underbrace{f_\theta(x) \in \mathbb{R}}_{\text{prediction}}.$$

The H entries of $h(x)$ form the **hidden layer**: they are computed inside the model, not observed or supplied as training targets.

In this lecture, depth one means one hidden layer, and fully connected means every input coordinate enters every h_j. The function σ is fixed; $\theta = \{a_j, w_j, b_j\}_{j=1}^H$ is fitted.

Three Common Activation Functions



ReLU is piecewise linear, GELU is smooth, and sigmoid is bounded with flat tails. Here Φ is the **standard normal CDF**, $\Phi(z) = \mathbb{P}(Z \le z)$ for $Z \sim \mathcal{N}(0,1)$, so GELU multiplies z by a smooth gate $\Phi(z) \in (0,1)$. Activation functions brings nonlinearity to NN. Without an activation, stacked affine layers collapse to one affine map.

Course illustration. GELU definition: Hendrycks and Gimpel (2016).

How the Parametric Solutions Differ

class	fitted parameters	solution method
degree-r polynomial	$r + 1$ coefficients	linear least squares
fixed-width network	all weights and biases	iterative gradient method

- Both classes have finitely many parameters once the architecture is fixed, so both are **parametric**.
- Polynomial regression is linear in its parameters: the normal equations apply, exactly as in Lecture 3.
- A network is nonconvex in its parameters and has no closed form.

Both are nonetheless fitted the same way in practice, as the next slide shows.

Fitting Any Parametric Model by Gradient Descent

derive

Write the class as $\mathcal{F} = \{f_\theta : \theta \in \mathbb{R}^p\}$. The empirical risk is the same squared loss as always,

$$\widehat{R}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2,$$

and differentiating through f_θ by the chain rule gives

$$\boxed{\nabla_\theta \widehat{R}_n(\theta) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))\,\nabla_\theta f_\theta(x_i)}$$

Fitting Any Parametric Model by Gradient Descent

derive

Write the class as $\mathcal{F} = \{f_\theta : \theta \in \mathbb{R}^p\}$. The empirical risk is the same squared loss as always,

$$\widehat{R}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2,$$

and differentiating through f_θ by the chain rule gives

$$\boxed{\nabla_\theta \widehat{R}_n(\theta) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))\,\nabla_\theta f_\theta(x_i)}$$

Gradient descent is then the iteration of Lecture 5, $\theta_{t+1} = \theta_t - \eta\,\nabla_\theta \widehat{R}_n(\theta_t)$, and a **minibatch** step replaces the sum over all n rows by a sum over a sampled subset B.

The Only Model-Specific Piece Is $\nabla_\theta f_\theta$

- **Basis expansion.** $f_\theta(x) = \theta^\top \phi(x)$ gives $\nabla_\theta f_\theta(x) = \phi(x)$, so the gradient is $-\frac{2}{n}\Phi^\top(y - \Phi\theta)$ — the least-squares gradient of Lecture 5, with Φ in place of X.
- **Network.** $\nabla_\theta f_\theta(x)$ is computed by **backpropagation**, the subject of Lecture 11.

The Only Model-Specific Piece Is $\nabla_\theta f_\theta$

- **Basis expansion.** $f_\theta(x) = \theta^\top \phi(x)$ gives $\nabla_\theta f_\theta(x) = \phi(x)$, so the gradient is $-\frac{2}{n}\Phi^\top(y - \Phi\theta)$ — the least-squares gradient of Lecture 5, with Φ in place of X.
- **Network.** $\nabla_\theta f_\theta(x)$ is computed by **backpropagation**, the subject of Lecture 11.

One optimizer, one loss, one update rule. Changing the model class changes only how $\nabla_\theta f_\theta$ is evaluated.

Summary: Parametric Regression

- **Assumption.** The regression function m lies in, or close to, a parametric family $\mathcal{F} = \{f_\theta : \theta \in \mathbb{R}^p\}$ fixed before seeing the data.
- **Objective.** The squared-loss empirical risk $\widehat{R}_n(\theta) = \frac{1}{n}\sum_i (y_i - f_\theta(x_i))^2$.
- **Bias.** Small exactly when m is close to $\mathcal{F}$. It is set by the choice of class, and no amount of data removes it.
- **Fitting.** Gradient descent on $\widehat{R}_n$. A closed form exists only in the special case that f_θ is linear in θ.

Summary: Parametric Regression

- **Assumption.** The regression function m lies in, or close to, a parametric family $\mathcal{F} = \{f_\theta : \theta \in \mathbb{R}^p\}$ fixed before seeing the data.
- **Objective.** The squared-loss empirical risk $\widehat{R}_n(\theta) = \frac{1}{n}\sum_i (y_i - f_\theta(x_i))^2$.
- **Bias.** Small exactly when m is close to $\mathcal{F}$. It is set by the choice of class, and no amount of data removes it.
- **Fitting.** Gradient descent on $\widehat{R}_n$. A closed form exists only in the special case that f_θ is linear in θ.

Next we drop the assumption of a fixed finite-parameter class altogether.

Outline

Estimating the Regression Function at a Point

The target has not changed: $m(x) = \mathbb{E}[Y \mid X = x]$.

- If many workers were exactly age 40, we would estimate $m(40)$ by averaging their wages.
- With a continuous feature no two rows share an x exactly, so there is nothing to average.
- So we **assume** m is continuous: workers near age 40 have a similar conditional mean.

Estimating the Regression Function at a Point

The target has not changed: $m(x) = \mathbb{E}[Y \mid X = x]$.

- If many workers were exactly age 40, we would estimate $m(40)$ by averaging their wages.
- With a continuous feature no two rows share an x exactly, so there is nothing to average.
- So we **assume** m is continuous: workers near age 40 have a similar conditional mean.

That assumption is the licence for everything below: replace "exactly at x" by "near x", and average those responses.

The Local Average

derive

Fix a radius $h > 0$ and collect the neighbours of the query,

$$N_h(x) = \{\, i : \, |x_i - x| \le h \,\},$$

then average their responses:

$$\boxed{\widehat{m}_h(x) = \frac{1}{|N_h(x)|} \sum_{i \in N_h(x)} y_i}$$

The Local Average

Fix a radius $h > 0$ and collect the neighbours of the query,

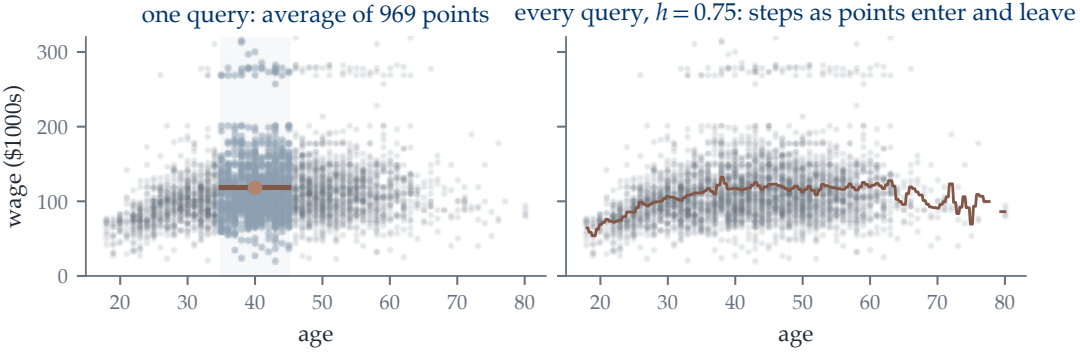
$$N_h(x) = \{\, i : \, |x_i - x| \le h \,\},$$

then average their responses:

$$\boxed{\widehat{m}_h(x) = \frac{1}{|N_h(x)|} \sum_{i \in N_h(x)} y_i}$$

- m is nearly constant across the window, so these y_i estimate about the same number;
- The noise averages out over the $|N_h(x)|$ points.

The Local Average in Pictures



Left: at age 40 with $h = 5$, the estimate is the mean of the 969 workers in the shaded band. Right: repeating this at every query traces a curve, which is rough because points enter and leave the window abruptly.

Course calculation from the ISLP `Wage` data.

The Bandwidth Trades Bias Against Variance

- **Large h.** Many points, so the noise averages away — but m is not constant across a wide window, so the estimate is biased toward the average level of a broad region.
- **Small h.** m is nearly constant across the window — but few points contribute, so the estimate is noisy.

The Bandwidth Trades Bias Against Variance

- **Large h.** Many points, so the noise averages away — but m is not constant across a wide window, so the estimate is biased toward the average level of a broad region.
- **Small h.** m is nearly constant across the window — but few points contribute, so the estimate is noisy.

The same trade-off as ridge in Lecture 4, in a new place: one knob moves bias and variance in opposite directions.

k-Nearest Neighbours

A fixed radius misbehaves where the data are uneven.

k-Nearest Neighbours

A fixed radius misbehaves where the data are uneven. Fix the count instead of the radius. Let $N_k(x)$ hold the k observations with the smallest $|x_i - x|$:

$$\widehat{m}_k(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

k-Nearest Neighbours

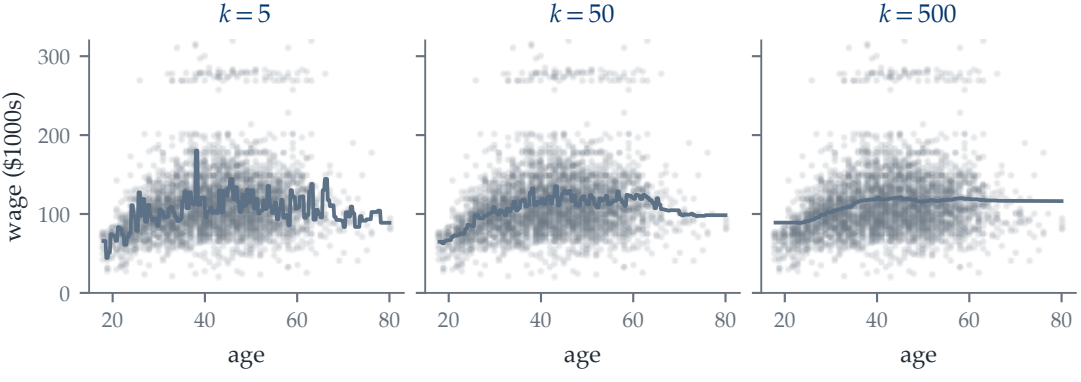
A fixed radius misbehaves where the data are uneven. Fix the count instead of the radius. Let $N_k(x)$ hold the k observations with the smallest $|x_i - x|$:

$$\widehat{m}_k(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

- $k = 1$ interpolates, returning the nearest response;
- $k = n$ returns the global mean $\bar{y}$ everywhere;
- In between, k plays the role h played.

Source: *K*-nearest neighbours regression, ISL §3.5.

k Controls Smoothness



Small k chases individual workers; large k flattens the rise and the decline. The neighbourhood is now defined by counting, so it widens automatically where the data thin out — past age 60, for instance.

Course fits to the ISLP `Wage` data.

Both Estimators Use a Hard Cutoff

- A point just inside the neighbourhood counts fully; one just outside counts not at all.
- As the query moves, points cross that edge and the estimate jumps — the roughness in the figure above.
- Every neighbour counts equally, whether it sits on top of the query or out at the boundary.

Both Estimators Use a Hard Cutoff

- A point just inside the neighbourhood counts fully; one just outside counts not at all.
- As the query moves, points cross that edge and the estimate jumps — the roughness in the figure above.
- Every neighbour counts equally, whether it sits on top of the query or out at the boundary.

The fix: let a point's influence fade with distance instead of switching off.

Soft Neighbourhoods Are Weighted Averages

Give observation i a weight $w_i(x) \geq 0$ that decreases with $|x_i - x|$, and take the weighted average

$$\widehat{m}(x) = \frac{\sum_{i=1}^{n} w_i(x)\, y_i}{\sum_{i=1}^{n} w_i(x)}$$

Soft Neighbourhoods Are Weighted Averages

Give observation i a weight $w_i(x) \geq 0$ that decreases with $|x_i - x|$, and take the weighted average

$$\widehat{m}(x) = \frac{\sum_{i=1}^{n} w_i(x)\, y_i}{\sum_{i=1}^{n} w_i(x)}$$

The two estimators so far are the special cases with 0/1 weights:

- Local average: $w_i(x) = \mathbf{1}\{|x_i - x| \leq h\}$;
- k-nearest neighbours: $w_i(x) = \mathbf{1}\{i \in N_k(x)\}$.

A weight that decays smoothly removes the jumps, because no point ever enters or leaves abruptly.

Kernel Regression

A **kernel** $K(u) \geq 0$ supplies the shape of the decay and a **bandwidth** $h > 0$ supplies its scale, giving $w_i(x) = K((x_i - x)/h)$ and the **Nadaraya–Watson** estimator

$$\widehat{m}_h(x) = \frac{\sum_i K\left(\frac{x_i - x}{h}\right) y_i}{\sum_i K\left(\frac{x_i - x}{h}\right)}$$

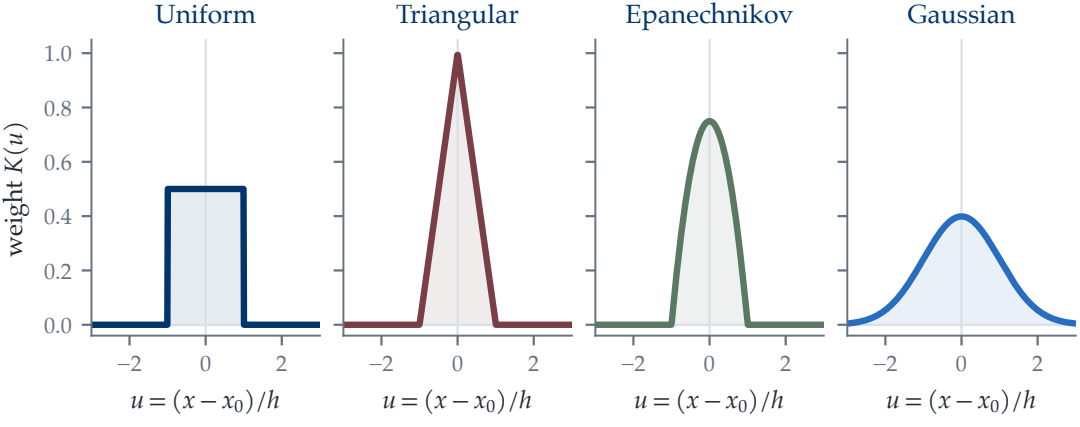
Kernel Regression

A **kernel** $K(u) \geq 0$ supplies the shape of the decay and a **bandwidth** $h > 0$ supplies its scale, giving $w_i(x) = K((x_i - x)/h)$ and the **Nadaraya–Watson** estimator

$$\widehat{m}_h(x) = \frac{\sum_i K\left(\frac{x_i - x}{h}\right) y_i}{\sum_i K\left(\frac{x_i - x}{h}\right)}$$

- There is no global coefficient vector: the weights are recomputed at every query;
- Scaling K by a positive constant changes nothing, since it cancels;
- The box kernel $K(u) = \mathbf{1}\{|u| \leq 1\}$ recovers the local average.

Common Kernel Choices



Course plots of four standard symmetric kernels; each is centered at the query after replacing u by $(x - x_0)/h$.

Kernel Regression as Weighted Least Squares

derive

The same estimator arrives from an optimization view. At the query x_0, fit a single constant a by distance-weighted squared error:

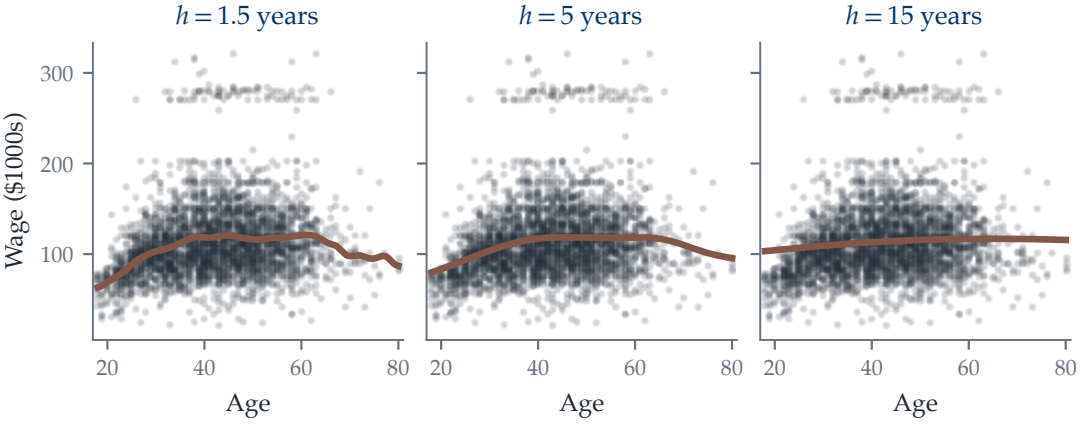
$$\hat{a}(x_0) = \arg\min_{a \in \mathbb{R}} \sum_{i=1}^{n} K\!\left(\frac{x_i - x_0}{h}\right)(y_i - a)^2.$$

Writing $K_i = K((x_i - x_0)/h)$, the first-order condition is

$$-2\sum_i K_i(y_i - a) = 0 \quad \Longrightarrow \quad \hat{a}(x_0) = \frac{\sum_i K_i y_i}{\sum_i K_i},$$

which is exactly $\widehat{m}_h(x_0)$.

Kernel Bandwidth



Small h follows local fluctuations; large h averages away the rise and decline. The kernel is Gaussian in all three panels, so only locality changes.

Course fits to the ISLP `Wage` data.

Local Linear Regression

Fitting a constant biases the estimate near a boundary, where the neighbours all lie on one side. Fit a local line instead:

$$(\hat{a}, \hat{b}) = \arg\min_{a,b} \sum_{i=1}^{n} K\left(\frac{x_i - x_0}{h}\right) [y_i - a - b(x_i - x_0)]^2,$$

and predict $\widehat{m}(x_0) = \hat{a}$, since $x_0 - x_0 = 0$.

Raising the local degree gives local polynomial regression. This is not the same as one global high-degree polynomial: the coefficients are refitted at every query.

Solving the Local Linear Fit

derive

It is a **weighted least squares** problem. Collect

$$Z = \begin{pmatrix} 1 & x_1 - x_0 \\ \vdots & \vdots \\ 1 & x_n - x_0 \end{pmatrix} \in \mathbb{R}^{n \times 2}, \qquad W = \operatorname{diag}(K_1, \dots, K_n), \qquad c = \begin{pmatrix} a \\ b \end{pmatrix},$$

so the objective is $(y - Zc)^\top W (y - Zc)$. Setting its gradient to zero,

$$Z^\top W (y - Zc) = 0 \quad \Longrightarrow \quad \boxed{\hat{c} = (Z^\top W Z)^{-1} Z^\top W y, \qquad \widehat{m}(x_0) = \hat{a}}$$

A 2×2 solve, repeated at every query. Taking Z to be a single column of ones recovers the Nadaraya–Watson formula, so the local constant fit is the same computation with one fewer column.

Local Methods Fail in High Dimensions

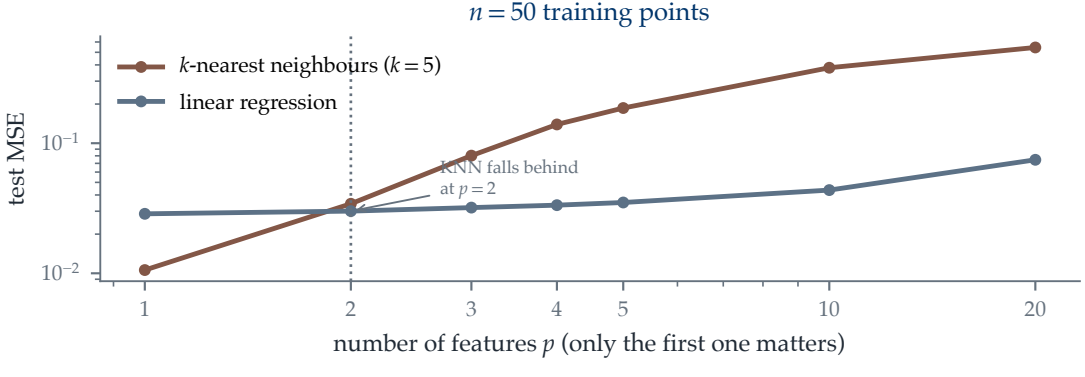
- Every method above needs neighbours: points close enough that m is nearly constant between them.
- In $[0,1]^d$ a window of side h holds a fraction h^d of the volume: 0.5 in one dimension, 0.03 in five, 0.001 in ten.
- Keeping a fixed number of points inside it therefore needs $n \propto h^{-d}$ — exponentially many observations.
- With n fixed the window must instead widen until it is no longer local, and the bias returns.

Local Methods Fail in High Dimensions

- Every method above needs neighbours: points close enough that m is nearly constant between them.
- In $[0,1]^d$ a window of side h holds a fraction h^d of the volume: 0.5 in one dimension, 0.03 in five, 0.001 in ten.
- Keeping a fixed number of points inside it therefore needs $n \propto h^{-d}$ — exponentially many observations.
- With n fixed the window must instead widen until it is no longer local, and the bias returns.

Parametric methods tend to win when there are few observations per feature.

Adding Irrelevant Features Destroys a Local Fit



The response depends on the first feature only; the rest are noise. With $n = 50$, k-nearest neighbours beats least squares at $p = 1$ and loses from $p = 3$ on. Its test error grows 51× by $p = 20$, against 2.6× for the linear fit.

Course simulation following ISL Figure 3.20; 60 replications per point.

Assumptions Buy Data

- **Parametric.** A strong assumption: m lies in a class fixed in advance, described by p numbers. Only those p numbers are estimated, so the data needed scales with p — not with d. If the assumption is wrong, the bias stays no matter how much data arrives.
- **Nonparametric.** A weak assumption: only that m is continuous. The class is enormous, so almost no bias — but the data needed grows exponentially in d.

Assumptions Buy Data

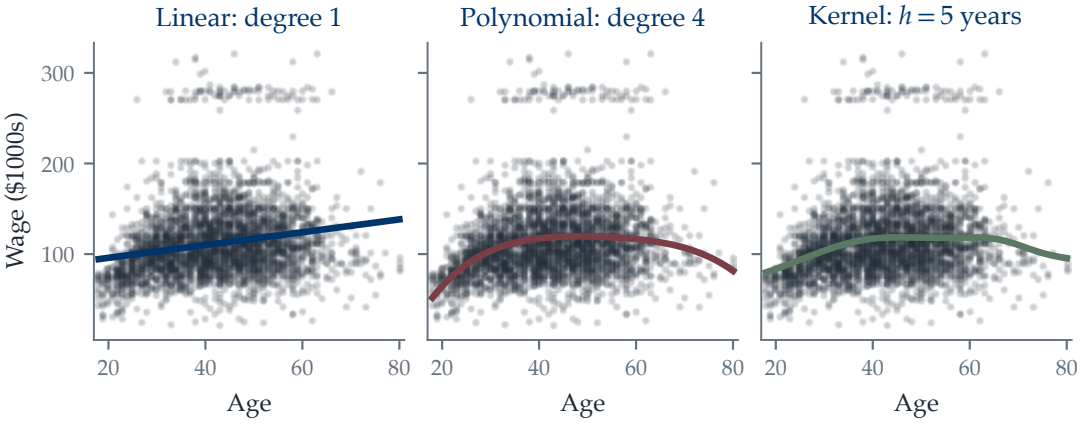
- **Parametric.** A strong assumption: m lies in a class fixed in advance, described by p numbers. Only those p numbers are estimated, so the data needed scales with p — not with d. If the assumption is wrong, the bias stays no matter how much data arrives.
- **Nonparametric.** A weak assumption: only that m is continuous. The class is enormous, so almost no bias — but the data needed grows exponentially in d.

More assumption, less data. Less assumption, more data. Feature expansions sit in between: a basis ϕ adds flexibility while keeping the model linear in its parameters.

Summary: Nonparametric Local Regression

- **Assumption.** No parametric form for m — only that m is continuous, so $m(x_i) \approx m(x)$ for x_i near x. That is what licenses using nearby y_i.
- **Estimator.** A weighted average $\widehat{m}(x) = \sum_i w_i(x) y_i / \sum_i w_i(x)$, the weight falling off with distance.
- **Bias and variance.** Bias is small when m is nearly constant over the neighbourhood; variance is small when many points contribute. One knob, h or k, trades them, chosen by cross-validation.
- **Fitting.** No parameters and no optimization: a formula evaluated afresh at each query, using all n rows.

Comparing Defined Model Families



The linear and degree-four curves are global least-squares fits. The kernel curve repeats a local constant fit at each age with $h = 5$ years.

Course fits to the ISLP `Wage` data.

Outline

The Fitted Model Is Itself Random

Everything so far treated the fit as fixed. It is not.

- The training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ is a random sample from the population.
- The fitted rule $\hat{f}_{\mathcal{D}}$ is computed from $\mathcal{D}$, so it is a random function: collect a fresh sample, run the same procedure, and a different rule comes out.
- Fix a query x_0. The prediction $\hat{f}_{\mathcal{D}}(x_0)$ is then a random variable.

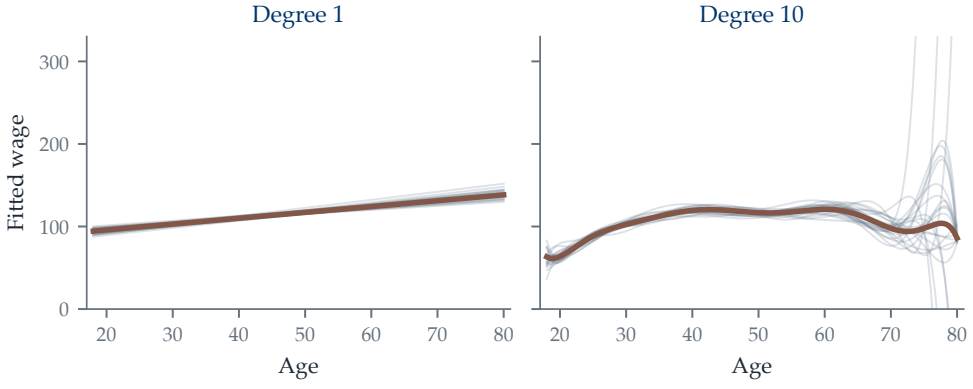
The Fitted Model Is Itself Random

Everything so far treated the fit as fixed. It is not.

- The training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ is a random sample from the population.
- The fitted rule $\hat{f}_{\mathcal{D}}$ is computed from $\mathcal{D}$, so it is a random function: collect a fresh sample, run the same procedure, and a different rule comes out.
- Fix a query x_0. The prediction $\hat{f}_{\mathcal{D}}(x_0)$ is then a random variable.

Two questions about any random variable: where is it centred, and how far does it move? Those are **bias** and **variance**.

The Same Procedure on Different Samples



Each curve is the same procedure refitted on a different random subset of 600 observed rows. The degree-1 curves nearly coincide; the degree-10 curves spread widely, especially near the boundaries. That spread illustrates the sample sensitivity measured by variance.

ISLP `Wage` data; 24 random subsets without replacement, seed 265.

Bias and Variance at a Point

Fix the query x_0 and write $\bar{f}(x_0) = \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x_0)]$ for the centre of that random variable. Then

$$\text{Bias}(x_0) = \bar{f}(x_0) - m(x_0), \qquad \text{Var}(x_0) = \mathbb{E}_{\mathcal{D}}[(\hat{f}_{\mathcal{D}}(x_0) - \bar{f}(x_0))^2].$$

Bias and Variance at a Point

Fix the query x_0 and write $\bar{f}(x_0) = \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x_0)]$ for the centre of that random variable. Then

$$\text{Bias}(x_0) = \bar{f}(x_0) - m(x_0), \qquad \text{Var}(x_0) = \mathbb{E}_{\mathcal{D}}[(\hat{f}_{\mathcal{D}}(x_0) - \bar{f}(x_0))^2].$$

- **Bias** asks whether the procedure aims at the right place;
- **Variance** asks how much it moves when the data change.

Neither is computable from one data set: both average over samples we never see. They describe the procedure, not the fit in front of us.

The Decomposition, Recalled from Lecture 4

For ridge we computed bias and variance exactly and watched one knob trade them. The same decomposition holds for any procedure: with $Y_0 = m(x_0) + \varepsilon_0$,

$$\mathbb{E}[(Y_0 - \hat{f}_{\mathcal{D}}(x_0))^2] = \underbrace{\sigma^2}_{\text{irreducible}} + \underbrace{\text{Bias}(x_0)^2}_{\text{systematic miss}} + \underbrace{\text{Var}(x_0)}_{\text{sample sensitivity}} .$$

The Decomposition, Recalled from Lecture 4

For ridge we computed bias and variance exactly and watched one knob trade them. The same decomposition holds for any procedure: with $Y_0 = m(x_0) + \varepsilon_0$,

$$\mathbb{E}[(Y_0 - \hat{f}_{\mathcal{D}}(x_0))^2] = \underbrace{\sigma^2}_{\text{irreducible}} + \underbrace{\text{Bias}(x_0)^2}_{\text{systematic miss}} + \underbrace{\text{Var}(x_0)}_{\text{sample sensitivity}} .$$

σ^2 belongs to the problem; bias and variance belong to the procedure.

Where the Trade Appeared in This Lecture

Every complexity knob met so far moves the two in opposite directions.

knob	raising it	seen in
degree r	bias $\downarrow$, variance $\uparrow$	degree 1 against degree 10
neighbours k	bias $\uparrow$, variance $\downarrow$	$k = 5$ against $k = 500$
bandwidth h	bias $\uparrow$, variance $\downarrow$	the three-panel bandwidth figure

Where the Trade Appeared in This Lecture

Every complexity knob met so far moves the two in opposite directions.

knob	raising it	seen in
degree r	bias $\downarrow$, variance $\uparrow$	degree 1 against degree 10
neighbours k	bias $\uparrow$, variance $\downarrow$	$k = 5$ against $k = 500$
bandwidth h	bias $\uparrow$, variance $\downarrow$	the three-panel bandwidth figure

One number to choose in each case, and no way to read it off the training data — which is what we take up next.

Outline

Every Model Here Has One Complexity Knob

- Polynomial regression: the degree r;
- k-nearest neighbours: the neighbour count k;
- Kernel regression: the bandwidth h;
- A network: the width H.

Every Model Here Has One Complexity Knob

- Polynomial regression: the degree r;
- k-nearest neighbours: the neighbour count k;
- Kernel regression: the bandwidth h;
- A network: the width H.

Training error always prefers the most flexible end — the largest degree, the smallest k or h. So none of these can be read off the training data.

Every Model Here Has One Complexity Knob

- Polynomial regression: the degree r;
- k-nearest neighbours: the neighbour count k;
- Kernel regression: the bandwidth h;
- A network: the width H.

Training error always prefers the most flexible end — the largest degree, the smallest k or h. So none of these can be read off the training data.

Choose each of them exactly as λ was chosen in Lecture 4: cross-validate on the training rows, take the value with the smallest $\widehat{R}_{\mathrm{CV}}$, refit, and touch the test set once.

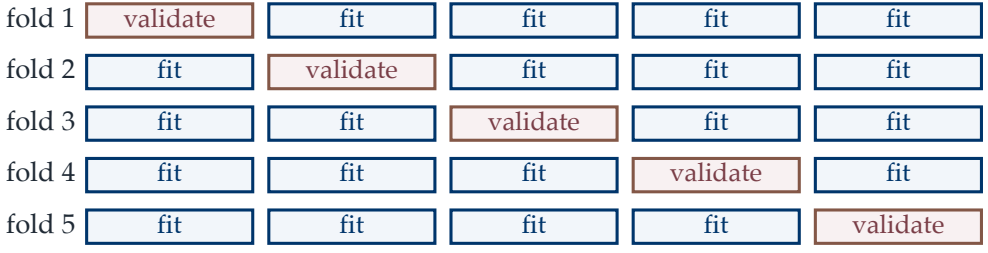
Choosing k or h by Cross-Validation

Nothing new is needed: this is the Lecture 4 procedure with λ replaced by k or h.

1. Split the training rows into K folds;
2. For each candidate value, fit on $K-1$ folds and score on the held-out fold, K times;
3. Average the scores into $\widehat{R}_{\mathrm{CV}}$;
4. Keep the value with the smallest $\widehat{R}_{\mathrm{CV}}$ and refit on all the training rows.

For a local method there is nothing to refit: "fitting" is storing the rows, and the chosen k or h simply enters the formula at prediction time.

Cross-Validation, Recalled from Lecture 4



each row is one split: fit on the four blue blocks, score on the red one

Each row is one split: fit on the four blue blocks, score on the red one. The same diagram governs the choice of k and h here.

Outside the Data, the Model Class Decides

Inside ages 18–80 the two fits agree and are equally well supported. Outside, they diverge completely, and the divergence is decided by the model class rather than by data: the polynomial continues its algebraic shape into negative wages, the kernel average flattens onto the boundary rows.

Course fits to the ISLP `Wage` data.

Prediction Is Supported Only Where the Data Is

- Cross-validation cannot separate those two curves: every fold it scores lies inside the observed range.
- So a good validation score certifies the fit there, and says nothing at all about age 10 or age 95.
- Asked about an input it never saw, the model still returns a number. Nothing in the fitting or the validation flags it as unsupported.

Prediction Is Supported Only Where the Data Is

- Cross-validation cannot separate those two curves: every fold it scores lies inside the observed range.
- So a good validation score certifies the fit there, and says nothing at all about age 10 or age 95.
- Asked about an input it never saw, the model still returns a number. Nothing in the fitting or the validation flags it as unsupported.

This is the simplest form of distribution shift: at prediction time the inputs come from a region the training inputs did not cover. Such inputs are called out of distribution, and no in-range score detects them.

Summary: Nonlinear Regression Models

Basis model. $\hat{f}(x) = \beta^\top \phi(x)$ is linear in β but nonlinear in x.

Kernel model.

$$\hat{f}_h(x_0) = \frac{\sum_i K((x_i - x_0)/h) y_i}{\sum_i K((x_i - x_0)/h)}.$$

Learned features. $\hat{f}(x) = w_2^\top \sigma(W_1 x + b_1) + b_2$ learns the basis jointly with the readout.

Summary: Bias, Variance, and Selection

At an input x_0 under squared loss,

$$\mathbb{E}\left[(Y_0 - \hat{f}_{\mathcal{D}}(x_0))^2\right] = \sigma^2 + \mathrm{Bias}(x_0)^2 + \mathrm{Var}(x_0).$$

Degree and bandwidth control the balance. Validation chooses among fitted procedures; held-out error and boundary behavior help diagnose why one choice may transfer better than another.

Next Lecture

Regression predicts a number. Many decisions instead require a label and a probability for each possible class.

Lecture 7 — Generative Classification: Bayes, LDA, and QDA models how each class could generate the observed features, applies Bayes' rule, and derives linear and quadratic decision boundaries.

Reading for this lecture: ISLP Chapter 7.