
CS/MATH 3414 — Numerical Methods

Lecture 6 Linear Regression

Virginia Tech Department of Computer Science, Fall 2026

Instructor: Sumeet Khatri

Date: September 29, 2026

1	From measurements to a linear model	2
2	Least squares and the normal equations	3
3	The pseudoinverse gives the regression solution	6
4	Fitting curves with polynomial regression	14
5	Choosing between a line and a higher-degree polynomial	16
5.1	Why training error alone cannot choose the degree	16
5.2	Measure performance on held-out data	16
5.3	Two datasets, two different degree choices	17
6	From least squares to neural networks	21
6.1	Least squares as supervised learning	21
6.2	From fixed features to learned features	22

Learning Objectives

- Turn a line or curve fitting problem into a matrix least-squares problem.
- Use the pseudoinverse to describe both unique and nonunique coefficient fits.
- Fit lines and polynomials, compute residuals, and interpret their patterns.
- Choose a polynomial degree using predictive performance and numerical reliability.
- Connect least-squares fitting to supervised learning, training loss, and prediction.

Where Lecture 5 leaves us

Lecture 5 studied $A\mathbf{x} \approx \mathbf{b}$ when an ordinary inverse is unavailable. It showed that $A^+\mathbf{b}$ is the minimum-norm least-squares solution, that $AA^+\mathbf{b}$ is the projection of the data onto $\text{Col}(A)$, and that small singular values can amplify errors. Here the unknown vector contains the coefficients of a fitted function. The modeling question is which columns to put into A .

Primary References

- [1] T. A. Driscoll and R. J. Braun, *Fundamentals of Numerical Computation*, Online python edition, See [Sec. 3.1: Fitting functions to data](#); [Sec. 3.2: The normal equations](#); [Sec. 3.3: The QR factorization](#); [Sec. 3.4: Computing QR factorizations](#); and [Sec. 7.3: Singular value decomposition](#), <https://fncbook.com/python/> (visited on 09/28/2026).

- [2] R. Penrose, “A generalized inverse for matrices”, *Proceedings of the Cambridge Philosophical Society* **51**, 406 (1955).
- [3] R. Penrose, “On best approximate solutions of linear matrix equations”, *Proceedings of the Cambridge Philosophical Society* **52**, 17 (1956).
- [4] G. James, D. Witten, T. Hastie, et al., *An Introduction to Statistical Learning, With applications in Python*, Chapters 2 (statistical learning), 3 (linear regression), 5 (resampling), 6 (model selection and regularization), and 7 (moving beyond linearity) (Springer, 2023), <https://www.statlearning.com/>.
- [5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning, Data mining, inference, and prediction*, 2nd ed., Chapters 2, 3, 5, and 7 cover supervised learning, regression, basis expansions, and model assessment (Springer, 2009), <https://hastie.su.domains/ElemStatLearn/main.html>.
- [6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Chapters 6 (feedforward networks and backpropagation) and 8 (optimization) (MIT Press, 2016), <https://www.deeplearningbook.org/>.
- [7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors”, *Nature* **323**, 533 (1986).
- [8] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, *Nature* **521**, 436 (2015).

1 From measurements to a linear model

Suppose we observe m pairs (t_i, b_i) , where t_i is an input and b_i is a measured response. We seek a function f that describes the relationship and can predict responses at new inputs. Measurements need not lie on an exact line or curve.

This construction is a standard numerical approach to fitting functions to data [1, Sec. 3.1]. For a straight line, write

$$f(t) = \alpha + \beta t, \quad b_i \approx \alpha + \beta t_i, \quad i = 1, \dots, m. \quad (1.1)$$

The intercept α is the predicted response at $t = 0$; the slope β is the predicted change in response per unit change in t . Stacking the equations gives

$$A\mathbf{x} \approx \mathbf{b}, \quad A = \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \quad (1.2)$$

The $m \times 2$ matrix A is called the *design matrix*. Its rows correspond to observations; its columns correspond to adjustable coefficients. The inputs t_i are known. The unknowns in this linear system are α and β .

Definition 1 (Linear regression in a chosen basis). Choose known functions $\phi_0, \dots, \phi_d$ and let

$$f(t) = \sum_{j=0}^d c_j \phi_j(t), \quad \mathbf{x} = (c_0, \dots, c_d)^\top, \quad A_{i,j+1} = \phi_j(t_i). \quad (1.3)$$

A *linear regression model* is linear in the unknown coefficients c_j . The functions ϕ_j may themselves be nonlinear in t .

Model	Fitted function	Row i of A
Constant	c_0	(1)
Line	$c_0 + c_1 t$	$(1, t_i)$
Quadratic	$c_0 + c_1 t + c_2 t^2$	$(1, t_i, t_i^2)$
Known-frequency wave	$c_0 + c_1 \sin t + c_2 \cos t$	$(1, \sin t_i, \cos t_i)$

By contrast, $f(t) = a e^{kt}$ with both a and k unknown is nonlinear in its parameters. Taking logarithms changes the residuals being minimized; it is a different fitting problem unless the error model justifies that transformation.

Checkpoint

Is fitting $c_0 + c_1 t + c_2 t^2$ linear regression? Yes: t_i^2 is known, so the unknown coefficients enter linearly. In this note, “line fit” means degree one; “linear regression” includes polynomial fits of any fixed degree.

2 Least squares and the normal equations

Definition 2 (Residual and sum of squared errors). For a coefficient vector $\mathbf{x}$, define the fitted values, residual, and sum of squared errors (SSE) by

$$\hat{\mathbf{b}} = A\mathbf{x}, \quad \mathbf{r} = \mathbf{b} - A\mathbf{x}, \quad \text{SSE}(\mathbf{x}) = \sum_{i=1}^m r_i^2 = \|\mathbf{b} - A\mathbf{x}\|_2^2. \quad (2.1)$$

An ordinary least-squares fit minimizes $\text{SSE}(\mathbf{x})$ over all coefficient vectors. Minimizing SSE/m or $\sqrt{\text{SSE}/m}$ gives the same coefficients.

Squaring prevents positive and negative residuals from canceling and penalizes large residuals more heavily. It also gives a tractable quadratic objective. This choice can make a fit sensitive to outliers.

Theorem 1 (Characterization of a least-squares fit). For every real $m \times p$ matrix A and data vector $\mathbf{b}$, a least-squares minimizer exists. A vector $\mathbf{x}_*$ is a minimizer if and only if

$$A^\top(\mathbf{b} - A\mathbf{x}_*) = 0, \quad \text{equivalently} \quad A^\top A\mathbf{x}_* = A^\top \mathbf{b}. \quad (2.2)$$

These are the *normal equations*. The fitted vector is always unique; the coefficient vector is unique exactly when A has full column rank.

Proof. Write $F(\mathbf{x}) = \text{SSE}(\mathbf{x})$. We will first identify a necessary condition for a minimum, then show why it is sufficient for this particular function, and finally establish existence and uniqueness.

From one-variable derivatives to the normal equations. Recall that if a differentiable function $g(s)$ has a local minimum at an interior point s_* , then $g'(s_*) = 0$. The converse is false: $g(s) = -s^2$ has derivative zero at $s = 0$, where it has a maximum. Thus finding a zero derivative is only the beginning of a minimization argument.

For F , hold all coordinates except x_j fixed. If $\mathbf{x}_*$ minimizes F , then $x_{*,j}$ minimizes this one-variable restriction, so $\partial F / \partial x_j = 0$ there. To calculate this derivative, write the i th residual as $r_i = b_i - \sum_{k=1}^p a_{ik}x_k$. The chain rule gives

$$\frac{\partial r_i}{\partial x_j} = -a_{ij}, \quad (2.3)$$

$$\frac{\partial F}{\partial x_j} = \sum_{i=1}^m 2r_i \frac{\partial r_i}{\partial x_j} = -2 \sum_{i=1}^m a_{ij}r_i = 2 \sum_{i=1}^m a_{ij} \left(\sum_{k=1}^p a_{ik}x_k - b_i \right). \quad (2.4)$$

The *gradient* collects these p partial derivatives into a column vector. Matrix multiplication now gives

$$\nabla F(\mathbf{x}) = 2A^\top(A\mathbf{x} - \mathbf{b}). \quad (2.5)$$

Consequently, every minimizer satisfies the normal equations (2.2).

For a line, this calculation takes a familiar form. With $F(\alpha, \beta) = \sum_i (b_i - \alpha - \beta t_i)^2$, it gives

$$\frac{\partial F}{\partial \alpha} = -2 \sum_i (b_i - \alpha - \beta t_i), \quad \frac{\partial F}{\partial \beta} = -2 \sum_i t_i (b_i - \alpha - \beta t_i). \quad (2.6)$$

Setting both derivatives to zero says that the residual is perpendicular to the constant column and the input column of the line's design matrix.

Why the stationary condition is sufficient here. Suppose $\mathbf{x}_*$ satisfies the normal equations, and set $\mathbf{r}_* = \mathbf{b} - A\mathbf{x}_*$. An arbitrary competing vector is $\mathbf{x}_* + \mathbf{h}$. Expanding its squared residual gives

$$\begin{aligned} F(\mathbf{x}_* + \mathbf{h}) &= \|\mathbf{r}_* - A\mathbf{h}\|_2^2 \\ &= \|\mathbf{r}_*\|_2^2 - 2\mathbf{r}_*^\top A\mathbf{h} + \|A\mathbf{h}\|_2^2 \\ &= F(\mathbf{x}_*) + \|A\mathbf{h}\|_2^2 \geq F(\mathbf{x}_*). \end{aligned} \quad (2.7)$$

The cross term vanishes because $\mathbf{r}_*^\top A\mathbf{h} = (A^\top \mathbf{r}_*)^\top \mathbf{h} = 0$. This proves a *global* minimum: no change in any combination of coordinates can lower F . The nonnegative final term supplies the justification that a zero gradient alone would not provide for a general function.

Existence, including deficient rank. The column space has an orthonormal basis $\mathbf{q}_1, \dots, \mathbf{q}_r$, where $r = \text{rank}(A)$. Its orthogonal projection of the data is

$$\mathbf{y}_* = \sum_{j=1}^r \mathbf{q}_j (\mathbf{q}_j^\top \mathbf{b}). \quad (2.8)$$

Since $\mathbf{y}_* \in \text{Col}(A)$, at least one coefficient vector satisfies $A\mathbf{x}_* = \mathbf{y}_*$. The vector $\mathbf{b} - \mathbf{y}_*$ is perpendicular to each basis vector, hence to every column of A . This $\mathbf{x}_*$ satisfies the normal equations and is therefore a minimizer. If $A = 0$, the sum is empty and equals zero; every coefficient vector then gives the same residual $\mathbf{b}$. No invertibility assumption was needed.

Finally, (2.7) shows that another coefficient vector has the same minimum objective value exactly when $A\mathbf{h} = 0$. Thus all minimizers differ by null-space vectors and produce the same fitted vector. Their coefficients are unique exactly when $\text{Null}(A) = \{0\}$, equivalently when A has full column rank. ■

The second-derivative viewpoint. The Hessian is the matrix of second partial derivatives. Differentiating once more gives

$$\nabla^2 F(\mathbf{x}) = 2A^\top A, \quad \mathbf{h}^\top \nabla^2 F(\mathbf{x}) \mathbf{h} = 2\|A\mathbf{h}\|_2^2 \geq 0. \quad (2.9)$$

This is the multivariable analogue of a nonnegative second derivative: along every line in coefficient space the objective curves upward or is flat. The objective is therefore *convex*, which means its value between two coefficient vectors lies below or on the straight line joining their objective values. Deficient rank allows flat directions; full column rank rules them out.

If A has full column rank, $A^\top A$ is positive definite because $\mathbf{h}^\top A^\top A \mathbf{h} = \|A\mathbf{h}\|_2^2 > 0$ for $\mathbf{h} \neq 0$. Consequently,

$$\mathbf{x}_* = (A^\top A)^{-1} A^\top \mathbf{b}. \quad (2.10)$$

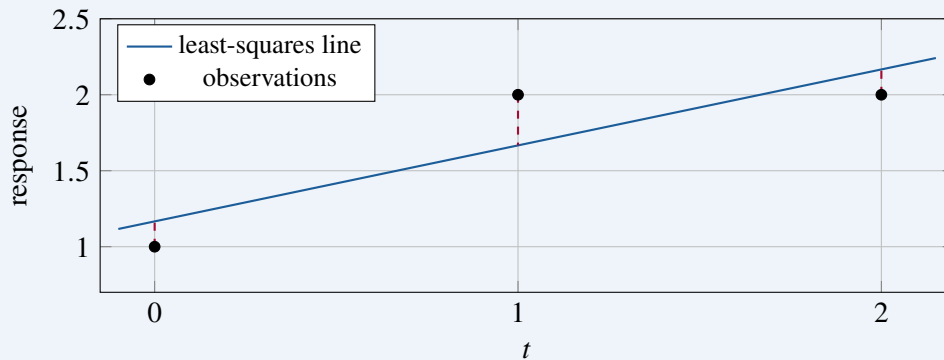
Example 1 (The three measurements from Lecture 5). Fit $f(t) = \alpha + \beta t$ to $(0, 1), (1, 2), (2, 2)$. The normal equations are

$$\begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}. \quad (2.11)$$

Subtracting the first equation from the second gives $2\beta = 1$, so

$$\hat{f}(t) = \frac{7}{6} + \frac{1}{2}t, \quad \hat{\mathbf{b}} = \begin{pmatrix} 7/6 \\ 5/3 \\ 13/6 \end{pmatrix}, \quad \mathbf{r} = \begin{pmatrix} -1/6 \\ 1/3 \\ -1/6 \end{pmatrix}, \quad \text{SSE} = \frac{1}{6}. \quad (2.12)$$

Both $\sum_i r_i = 0$ and $\sum_i t_i r_i = 0$ hold. At $t = 3/2$, the prediction is $23/12$. The residuals shown below are vertical distances with signs: positive means the observed response is above the fitted line.



A direct formula using centered data. Let $\bar{t} = m^{-1} \sum_i t_i$ and $\bar{b} = m^{-1} \sum_i b_i$. The first normal equation gives $\alpha = \bar{b} - \beta \bar{t}$. Substitution into the second gives

$$\beta = \frac{\sum_i (t_i - \bar{t})(b_i - \bar{b})}{\sum_i (t_i - \bar{t})^2}, \quad \alpha = \bar{b} - \beta \bar{t}. \quad (2.13)$$

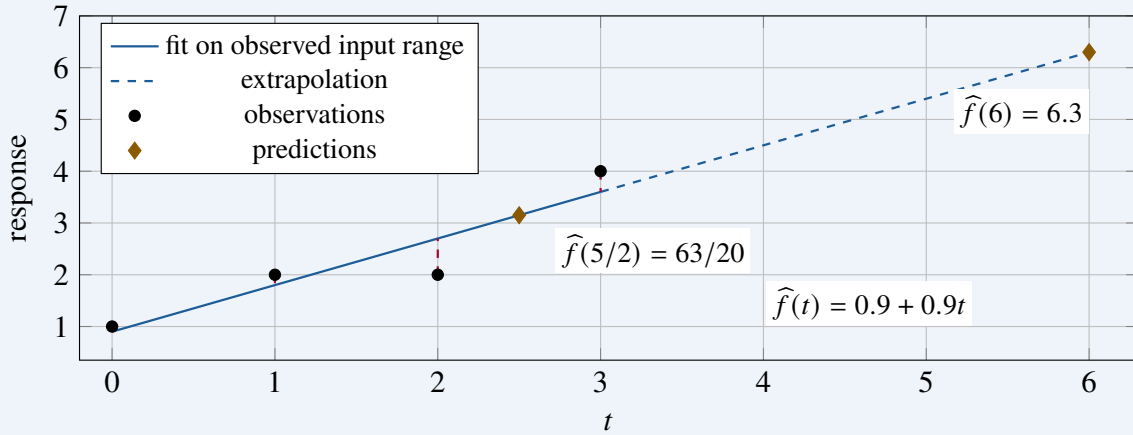
The denominator is positive exactly when at least two inputs are distinct. Every least-squares line with an intercept passes through $(\bar{t}, \bar{b})$.

Example 2 (A calibration line). Suppose the input levels are $t = (0, 1, 2, 3)$ and the measured responses are $\mathbf{b} = (1, 2, 2, 4)^\top$. Here $\bar{t} = 3/2$, $\bar{b} = 9/4$, and the numerator and denominator in (2.13) are $9/2$ and 5 . Thus

$$\hat{f}(t) = \frac{9}{10} + \frac{9}{10}t, \quad \mathbf{r} = \frac{1}{10}(1, 2, -7, 4)^\top, \quad \text{SSE} = \frac{7}{10}. \quad (2.14)$$

At $t = 5/2$, predict $63/20 = 3.15$ response units. At $t = 6$, the formula gives 6.3 , but this is *extrapolation*: there are no measurements beyond $t = 3$ supporting the continued linear trend.

The plot distinguishes predictions within the observed input range from extrapolation beyond it. The vertical residual segments compare the measured responses with the fitted line; the predicted values at $t = 5/2$ and $t = 6$ are evaluations of that same line.



3 The pseudoinverse gives the regression solution

Lecture 5 defined the Moore–Penrose pseudoinverse through the compact SVD. The original papers of Penrose establish the generalized inverse and its connection to best approximate solutions [2, 3]. For a matrix of rank $r > 0$,

$$A = U_r \Sigma_r V_r^\top, \quad A^+ = V_r \Sigma_r^{-1} U_r^\top, \quad \Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r), \quad \sigma_j > 0. \quad (3.1)$$

The columns of U_r and V_r are orthonormal. For $A = 0$, define $A^+ = 0$ with dimensions reversed.

Theorem 2 (Regression through the pseudoinverse). For every design matrix $A \in \mathbb{R}^{m \times p}$,

$$\mathbf{x}_* = A^+ \mathbf{b} = \sum_{j=1}^r \frac{\mathbf{u}_j^\top \mathbf{b}}{\sigma_j} \mathbf{v}_j \quad (3.2)$$

is the unique minimum-norm least-squares coefficient vector. All least-squares coefficients, their common fitted values, and their residual are

$$\hat{\mathbf{b}} = H \mathbf{b}, \quad H = A A^+ = U_r U_r^\top, \quad (3.3)$$

$$\mathbf{r}_* = (\mathbb{I}_m - H) \mathbf{b}. \quad (3.4)$$

Here H is the orthogonal projection onto $\text{Col}(A)$, often called the *hat matrix*.

Proof. The design matrix $A \in \mathbb{R}^{m \times p}$ and the measured response vector $\mathbf{b} \in \mathbb{R}^m$ are fixed. The unknown is the coefficient vector $\mathbf{x} \in \mathbb{R}^p$. The least-squares problem is

$$\min_{\mathbf{x} \in \mathbb{R}^p} F(\mathbf{x}), \quad F(\mathbf{x}) = \|\mathbf{b} - A\mathbf{x}\|_2^2 = \sum_{i=1}^m \left(b_i - \sum_{k=1}^p A_{ik}x_k \right)^2. \quad (3.5)$$

Thus we vary the entries of $\mathbf{x}$ to make the squared residual as small as possible. We are not, at this stage, minimizing $\|\mathbf{x}\|_2$.

If $A = 0$, then $F(\mathbf{x}) = \|\mathbf{b}\|_2^2$ for every $\mathbf{x}$. Every coefficient vector is therefore a least-squares minimizer, and $\mathbf{x} = 0 = A^+\mathbf{b}$ is the unique one with smallest norm. The fitted vector is zero and the residual is $\mathbf{b}$, as claimed. For the remainder of the proof, suppose $r = \text{rank}(A) > 0$.

Step 1: Differentiate with respect to the unknown coefficients.

Recall from calculus that a differentiable function must have zero derivative at an interior minimum. Here there are p unknown coordinates, so every partial derivative must vanish. For $k = 1, \dots, p$, the chain rule gives

$$\frac{\partial F}{\partial x_k} = 2 \sum_{i=1}^m A_{ik} \left(\sum_{\ell=1}^p A_{i\ell}x_\ell - b_i \right). \quad (3.6)$$

Collecting these derivatives into the gradient gives

$$\nabla F(\mathbf{x}) = 2A^\top(A\mathbf{x} - \mathbf{b}). \quad (3.7)$$

Consequently, every least-squares minimizer must satisfy

$$A^\top A\mathbf{x} = A^\top \mathbf{b}. \quad (3.8)$$

These are equations for the unknown vector $\mathbf{x}$. A zero gradient alone does not yet prove that we have a minimum; we will establish that in Step 3.

Step 2: Solve these equations using the SVD.

Write the compact SVD as

$$A = U_r \Sigma_r V_r^\top, \quad U_r^\top U_r = V_r^\top V_r = \mathbb{1}_r, \quad \Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r), \quad \sigma_j > 0. \quad (3.9)$$

Here U_r is $m \times r$, V_r is $p \times r$, and Σ_r is an invertible $r \times r$ diagonal matrix. Substitution into (3.8) gives

$$V_r \Sigma_r^2 V_r^\top \mathbf{x} = V_r \Sigma_r U_r^\top \mathbf{b}. \quad (3.10)$$

Multiplying first by $V_r^\top$ and then by Σ_r^{-2} yields

$$V_r^\top \mathbf{x} = \Sigma_r^{-1} U_r^\top \mathbf{b}. \quad (3.11)$$

We have now determined the coordinates of $\mathbf{x}$ along the columns of V_r . In particular, one vector satisfying this condition is

$$\mathbf{x}_* = V_r \Sigma_r^{-1} U_r^\top \mathbf{b}, \quad (3.12)$$

because multiplying this vector by $V_r^\top$ recovers the right-hand side of (3.11). By the SVD definition of the pseudoinverse,

$\mathbf{x}_* = A^+ \mathbf{b}.$

(3.13)

Thus the pseudoinverse formula has emerged from solving the equations obtained by differentiating the least-squares objective. We have not assumed that A or $A^\top A$ is invertible.

Step 3: Prove that this vector actually minimizes the residual.

Set $\mathbf{r}_* = \mathbf{b} - A\mathbf{x}_*$. Since $\mathbf{x}_*$ satisfies the normal equations, $A^\top \mathbf{r}_* = 0$. Every competing coefficient vector can be written as $\mathbf{x}_* + \mathbf{h}$, where $\mathbf{h} \in \mathbb{R}^p$. Expanding its objective value gives

$$\begin{aligned} F(\mathbf{x}_* + \mathbf{h}) &= \|\mathbf{b} - A(\mathbf{x}_* + \mathbf{h})\|_2^2 \\ &= \|\mathbf{r}_* - A\mathbf{h}\|_2^2 \\ &= \|\mathbf{r}_*\|_2^2 - 2\mathbf{h}^\top A^\top \mathbf{r}_* + \|A\mathbf{h}\|_2^2 \\ &= F(\mathbf{x}_*) + \|A\mathbf{h}\|_2^2 \\ &\geq F(\mathbf{x}_*). \end{aligned} \tag{3.14}$$

The cross term vanishes because $A^\top \mathbf{r}_* = 0$. This proves a global minimum: no choice of the coefficient vector gives a smaller squared residual than $\mathbf{x}_* = A^+ \mathbf{b}$.

Moreover, equality holds exactly when $A\mathbf{h} = 0$. Therefore all least-squares minimizers are precisely

$$\mathbf{x} = A^+ \mathbf{b} + \mathbf{z}, \quad \mathbf{z} \in \text{Null}(A). \tag{3.15}$$

Their coefficients may differ, but their fitted vectors agree, because $A\mathbf{z} = 0$.

Step 4: Among these minimizers, select the smallest coefficient norm.

The least-squares minimization is now complete. We next ask a separate question: among the vectors that attain this minimum residual, which has the smallest Euclidean norm?

If $\mathbf{z} \in \text{Null}(A)$, then the SVD gives

$$0 = A\mathbf{z} = U_r \Sigma_r V_r^\top \mathbf{z}. \tag{3.16}$$

Multiplying by $U_r^\top$ and then by Σ_r^{-1} shows that $V_r^\top \mathbf{z} = 0$. Since $\mathbf{x}_* = V_r \Sigma_r^{-1} U_r^\top \mathbf{b}$ is a linear combination of the columns of V_r , it follows that $\mathbf{x}_*^\top \mathbf{z} = 0$. Hence

$$\|\mathbf{x}_* + \mathbf{z}\|_2^2 = \|\mathbf{x}_*\|_2^2 + \|\mathbf{z}\|_2^2 \geq \|\mathbf{x}_*\|_2^2. \tag{3.17}$$

Equality holds only when $\mathbf{z} = 0$. Thus $A^+ \mathbf{b}$ is the unique coefficient vector of smallest norm *among all least-squares minimizers*. This additional selection never increases the residual.

Step 5: Identify the fitted vector and residual.

For every least-squares minimizer,

$$\begin{aligned} \widehat{\mathbf{b}} &= A(A^+ \mathbf{b} + \mathbf{z}) = AA^+ \mathbf{b}, \\ H &= AA^+ = U_r \Sigma_r V_r^\top V_r \Sigma_r^{-1} U_r^\top = U_r U_r^\top. \end{aligned} \tag{3.18}$$

The columns of U_r form an orthonormal basis for $\text{Col}(A)$. Consequently, $H\mathbf{b}$ lies in $\text{Col}(A)$, while

$$U_r^\top (\mathbf{b} - H\mathbf{b}) = U_r^\top \mathbf{b} - (U_r^\top U_r) U_r^\top \mathbf{b} = 0. \tag{3.19}$$

Thus $\mathbf{b} - H\mathbf{b}$ is perpendicular to $\text{Col}(A)$, proving that H is its orthogonal projection matrix. In particular,

$$\widehat{\mathbf{b}} = H\mathbf{b}, \quad \mathbf{r}_* = (\mathbb{I}_m - H)\mathbf{b}. \tag{3.20}$$

Both vectors are unique, even when the least-squares coefficient vector is not. ■

Example 3. For Example 1, in Lecture 5 we computed

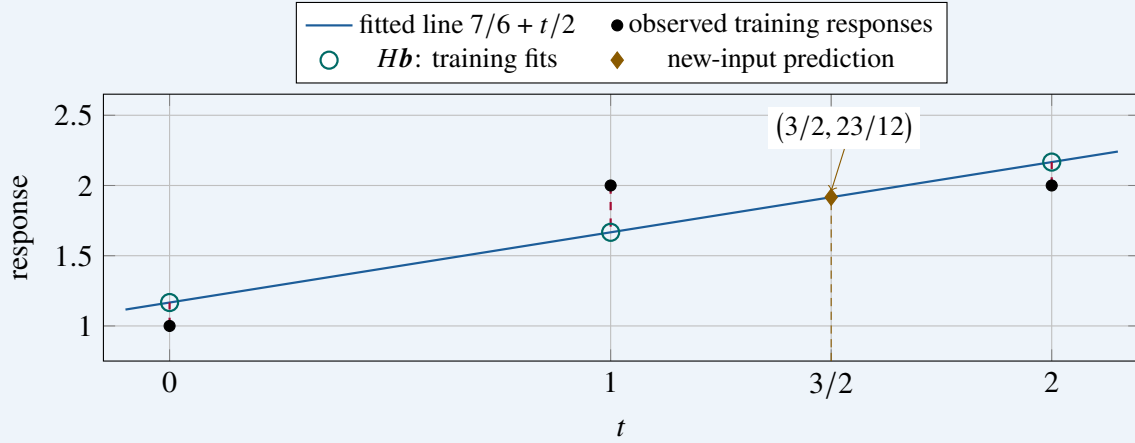
$$A^+ = \frac{1}{6} \begin{pmatrix} 5 & 2 & -1 \\ -3 & 0 & 3 \end{pmatrix}, \quad H = \frac{1}{6} \begin{pmatrix} 5 & 2 & -1 \\ 2 & 2 & 2 \\ -1 & 2 & 5 \end{pmatrix}. \quad (3.21)$$

Multiplying A^+ by $(1, 2, 2)^\top$ gives $(7/6, 1/2)^\top$. Multiplying H by the same data gives the fitted responses. If new responses are measured at the *same inputs* 0, 1, 2, the same two matrices apply. For a new input t_* , the predicted response is instead

$$\hat{f}(t_*) = \begin{pmatrix} 1 & t_* \end{pmatrix} A^+ \mathbf{b}. \quad (3.22)$$

The hat matrix maps responses to fitted values at the training inputs; evaluation of the fitted function gives predictions elsewhere.

The plot separates the observed responses from the fitted training values. The prediction $\hat{f}(3/2) = 23/12$ lies on the same fitted line at a new input; no response measured at $t = 3/2$ is needed to locate it.



Key Idea

A^+ returns coefficients; AA^+ returns fitted training values. The pseudoinverse makes the best fit within a specified model. It does not certify that the model predicts well.

Example 4 (Repeated inputs and an undetermined slope). Suppose that we measure the response three times at the *same* input $t = 1$, obtaining $\mathbf{b} = (1, 2, 3)^\top$. We fit a line $f(t) = \alpha + \beta t$, with coefficient vector $\mathbf{x} = (\alpha, \beta)^\top$. Since each row of the design matrix is $(1, t_i)$, we have

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad A\mathbf{x} = \begin{pmatrix} \alpha + \beta \\ \alpha + \beta \\ \alpha + \beta \end{pmatrix}. \quad (3.23)$$

The columns of A are identical, so $\text{rank}(A) = 1 < 2$. Although there are three measurements and only two unknown coefficients, the measurements all concern the *same value* $f(1) = \alpha + \beta$. They contain no information about how the response changes as t changes.

What does least squares determine? Let $q = \alpha + \beta$ denote the common predicted response. The least-squares objective depends only on q :

$$\begin{aligned} L(\alpha, \beta) &= (1 - q)^2 + (2 - q)^2 + (3 - q)^2 \\ &= 3q^2 - 12q + 14 = 3(q - 2)^2 + 2. \end{aligned} \quad (3.24)$$

The last expression is smallest when $q = 2$, the mean of the measured responses. Equivalently, calculus gives $dL/dq = 6(q - 2) = 0$ and $d^2L/dq^2 = 6 > 0$. This determines the *sum* $\alpha + \beta = 2$, not the two coefficients separately. The normal equations express the same fact:

$$\begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \end{pmatrix}. \quad (3.25)$$

Both rows give $3\alpha + 3\beta = 6$; there is only one independent equation. Every least-squares line therefore has

$$\widehat{\mathbf{b}} = (2, 2, 2)^\top, \quad \mathbf{r} = \mathbf{b} - \widehat{\mathbf{b}} = (-1, 0, 1)^\top, \quad \text{SSE} = 2. \quad (3.26)$$

No line can reproduce all three measured responses exactly: a function has just one value at $t = 1$.

Which coefficients give that same best fit? Choose any slope $s \in \mathbb{R}$ and set the intercept to $2 - s$. All least-squares solutions and their functions are

$$\mathbf{x}_s = \begin{pmatrix} 2 - s \\ s \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + (s - 1) \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad f_s(t) = 2 + s(t - 1). \quad (3.27)$$

The vector $(-1, 1)^\top$ lies in $\text{Null}(A)$: increasing the slope while decreasing the intercept by the same amount leaves every training prediction unchanged. For example,

Fitted function	Slope s	At $t = 1$ (training)	At $t = 0$ (new)
$f(t) = 2$	0	2	2
$f(t) = 1 + t$	1	2	1
$f(t) = 2t$	2	2	0

All three functions have the same SSE, yet disagree away from $t = 1$. More generally, at any fixed $t_* \neq 1$, the quantity $2 + s(t_* - 1)$ can take any real value as s varies. The training data alone cannot choose among these predictions.

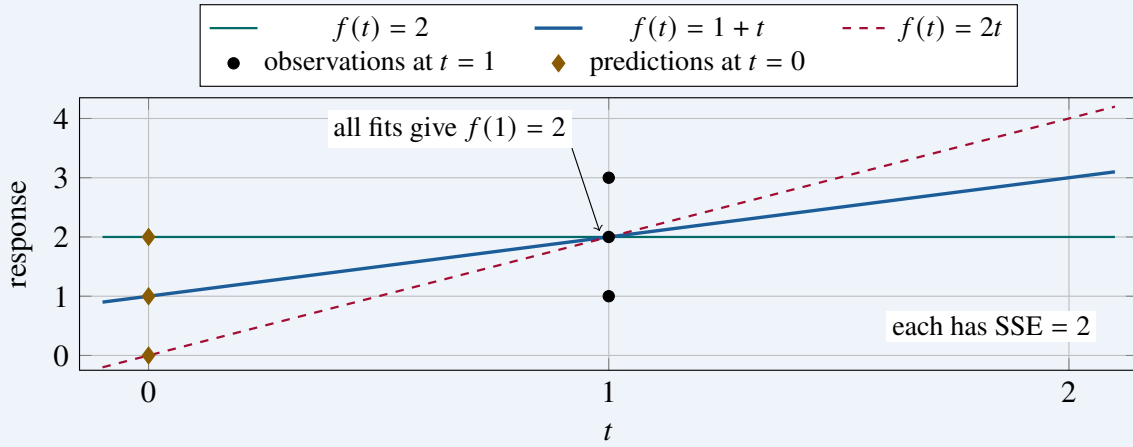
What additional choice does the pseudoinverse make? Among these equally good fits, it selects the coefficient vector of smallest Euclidean norm. Here

$$\|\mathbf{x}_s\|_2^2 = (2 - s)^2 + s^2 = 2(s - 1)^2 + 2, \quad (3.28)$$

which is uniquely minimized at $s = 1$. Accordingly,

$$A^+ = \frac{1}{6} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad A^+ \mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \hat{f}(t) = 1 + t. \quad (3.29)$$

This is a unique *minimum-norm selection*, not a slope uniquely established by the measurements. Repeating measurements at $t = 1$ can refine the estimate of the response there, but it cannot reveal a slope; measurements at a second distinct input are needed to identify both line coefficients by least squares. The three lines in the plot all pass through the common fitted point $(1, 2)$ and have the same residuals at the observed input. Their different intercepts show directly why predictions at $t = 0$ are not determined by these training data. The pseudoinverse selects the line $1 + t$ in the original coefficient coordinates.



The distinction between a unique fitted data vector and a unique fitted function matters. For a general basis, the change in prediction caused by a null-space vector $\mathbf{z}$ is

$$\sum_{j=0}^d z_j \phi_j(t_*). \quad (3.30)$$

It is zero at every training input, but may be nonzero elsewhere.

Example 5 (Minimum norm depends on the coefficient coordinates). Use the same observations as in Example 4, but replace the feature functions $(1, t)$ by $(1, 2t)$. We now write

$$f(t) = a + 2ct, \quad \alpha = a, \quad \beta = 2c. \quad (3.31)$$

This is still the family of *all lines*: every line $\alpha + \beta t$ is represented by $a = \alpha$ and $c = \beta/2$. We have changed how the coefficients describe a function, without changing the observations or the set of candidate functions.

Fit the data in the new coordinates. At $t = 1$, the predicted response is $a + 2c$, so the new design matrix and least-squares objective are

$$\tilde{A} = \begin{pmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{pmatrix}, \quad \tilde{L}(a, c) = 3(a + 2c - 2)^2 + 2. \quad (3.32)$$

Thus the least-squares condition is $a + 2c = 2$, with minimum SSE still equal to 2. In particular, all least-squares solutions still predict $(2, 2, 2)^\top$ at the three training inputs. The change of coordinates has not resolved the missing information about the slope.

Apply the minimum-norm rule in these coordinates. The pseudoinverse of $\tilde{A}$ minimizes $a^2 + c^2$ among the least-squares solutions. Substitute $a = 2 - 2c$ to obtain

$$a^2 + c^2 = (2 - 2c)^2 + c^2 = 5c^2 - 8c + 4 = 5\left(c - \frac{4}{5}\right)^2 + \frac{4}{5}. \quad (3.33)$$

The minimum occurs at $c = 4/5$, and hence $a = 2/5$. Alternatively, setting the derivative $10c - 8$ equal to zero gives the same result. Direct multiplication by the new pseudoinverse confirms it:

$$\tilde{A}^+ = \frac{1}{15} \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{pmatrix}, \quad \tilde{A}^+ \mathbf{b} = \begin{pmatrix} 2/5 \\ 4/5 \end{pmatrix}. \quad (3.34)$$

The selected function is therefore

$$\hat{f}_{\text{new}}(t) = \frac{2}{5} + \frac{8}{5}t. \quad (3.35)$$

At $t = 1$, this gives 2, exactly as the previous selection $\hat{f}_{\text{old}}(t) = 1 + t$ does. At $t = 0$, however, their predictions are $2/5$ and 1, respectively. Their difference is

$$\hat{f}_{\text{new}}(t) - \hat{f}_{\text{old}}(t) = \frac{3}{5}(t - 1), \quad (3.36)$$

so they agree at the training input and disagree at every other input.

Why is there no contradiction? The norm being minimized has changed. The original squared coefficient norm, expressed in the new coordinates, is

$$\alpha^2 + \beta^2 = a^2 + 4c^2, \quad (3.37)$$

whereas the new pseudoinverse minimizes $a^2 + c^2$. These are different measures of coefficient size. Equivalently, $a^2 + c^2 = \alpha^2 + \beta^2/4$, so the new rule gives less weight to a given slope in the original coordinates.

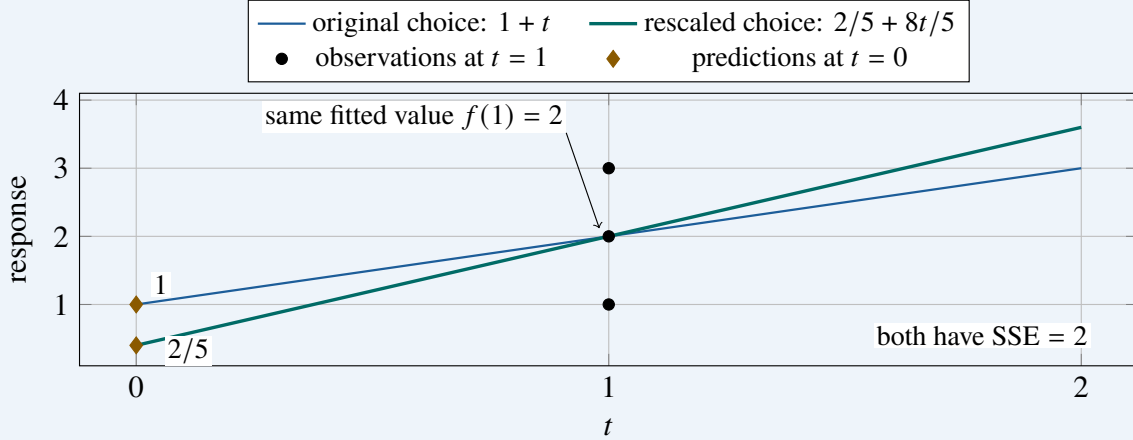
If we keep the *original* norm while merely rewriting the coefficients, we instead minimize

$$a^2 + 4c^2 = (2 - 2c)^2 + 4c^2 = 8\left(c - \frac{1}{2}\right)^2 + 2. \quad (3.38)$$

This selects $c = 1/2$ and $a = 1$, recovering the same function $1 + t$. Thus the different prediction arises from a different minimum-norm criterion among equally good training fits. The measurements

themselves favor neither selection. When least-squares coefficients are not unique, the basis and units must be specified before “the minimum-norm solution” has a definite meaning.

The plot compares the two minimum-norm selections as functions of the original input. They agree at the training input $t = 1$, but their intercepts are 1 and $2/5$. The changed coefficient norm selects a different line from the same family of least-squares fits.



Notions of uniqueness

For a fixed design matrix A and response vector $\mathbf{b}$, distinguish the following questions.

1. **Are the fitted values at the training inputs unique? Always.** Every least-squares minimizer gives the same vector

$$\hat{\mathbf{b}} = A\mathbf{x} = AA^+\mathbf{b}. \quad (3.39)$$

This is the unique orthogonal projection of $\mathbf{b}$ onto $\text{Col}(A)$. Consequently the residual and minimum SSE are also unique. The fitted values need not equal the measured responses: in Example 4, the unique fitted vector is $(2, 2, 2)^\top$, although the observations are $(1, 2, 3)^\top$.

2. **Are the coefficients unique? Exactly when A has full column rank.** All least-squares coefficient vectors have the form

$$\mathbf{x} = A^+\mathbf{b} + \mathbf{z}, \quad \mathbf{z} \in \text{Null}(A). \quad (3.40)$$

If $\text{Null}(A) = \{0\}$, there is only one choice. Otherwise there are infinitely many: a nonzero null-space direction changes the coefficients without changing $A\mathbf{x}$. Having more observations than coefficients does not by itself ensure uniqueness; the columns must also be linearly independent. The repeated inputs in Example 4 illustrate this distinction.

3. **Is a prediction at a new input unique? It depends on the input.** Let $\boldsymbol{\phi}(t_*) = (\phi_0(t_*), \dots, \phi_{p-1}(t_*))^\top$. Adding a null-space vector changes the predicted response by $\boldsymbol{\phi}(t_*)^\top \mathbf{z}$. Thus every least-squares minimizer agrees at t_* if and only if

$$\boldsymbol{\phi}(t_*)^\top \mathbf{z} = 0 \quad \text{for every } \mathbf{z} \in \text{Null}(A). \quad (3.41)$$

Equivalently, $\boldsymbol{\phi}(t_*)$ must lie in the row space $\text{Col}(A^\top)$. This holds automatically at every training input, but may fail elsewhere. In Example 4, only the prediction at $t_* = 1$ is fixed across all minimizing lines.

Recap (continued)

A unique coefficient vector determines a unique function throughout its domain. The converse holds for linearly independent basis functions, such as $1, t, \dots, t^d$, but can fail for redundant features. For example, the features 1 and 1 give $f(t) = c_0 + c_1$: different coefficient pairs with the same sum describe exactly the same constant function everywhere.

4. **Is the minimum-norm least-squares choice unique? Always, in fixed coordinates.** The pseudoinverse selects $\mathbf{x}_* = A^+ \mathbf{b}$ uniquely from the set of minimizers. This is an additional rule when A is rank deficient; it does not mean the data determine all the coefficients. Example 5 shows that rescaling a feature can change the selected function because an unweighted Euclidean norm in the new coordinates measures coefficient size differently.

With full column rank, there is no freedom left for a minimum-norm rule to resolve: an invertible change of basis only changes the coordinates of the same fitted function. With deficient rank, the fitted training values remain unchanged under such a basis change, but the minimum-norm function can change. Finally, uniqueness describes whether the stated fitting problem has one answer; it does not guarantee that the answer predicts new responses accurately.

4 Fitting curves with polynomial regression

For a polynomial of degree at most d , take $\phi_j(t) = t^j$. Then

$$f_d(t) = c_0 + c_1 t + \dots + c_d t^d, \quad A_d = \begin{pmatrix} 1 & t_1 & t_1^2 & \dots & t_1^d \\ 1 & t_2 & t_2^2 & \dots & t_2^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_m & t_m^2 & \dots & t_m^d \end{pmatrix}. \quad (4.1)$$

This is a Vandermonde design matrix. The coefficient vector is again $\mathbf{x}_* = A_d^+ \mathbf{b}$. The least-squares machinery does not change when the curve bends.

Theorem 3 (When polynomial coefficients are unique). For the basis $1, t, \dots, t^d$, A_d has full column rank if and only if the observed inputs contain at least $d + 1$ distinct values.

Proof. Recall that A_d has full column rank exactly when $A_d \mathbf{z} = 0$ has only the zero solution. To interpret this condition, associate each coefficient vector $\mathbf{z} = (z_0, \dots, z_d)^\top$ with the polynomial

$$p(t) = z_0 + z_1 t + \dots + z_d t^d, \quad A_d \mathbf{z} = (p(t_1), \dots, p(t_m))^\top. \quad (4.2)$$

Thus $\mathbf{z} \in \text{Null}(A_d)$ precisely when p vanishes at every observed input.

Suppose first that there are at least $d + 1$ distinct inputs. A nonzero polynomial of degree at most d has at most d distinct roots: each distinct root contributes a linear factor. Therefore a polynomial p with $A_d \mathbf{z} = 0$ must be the zero polynomial. All its coefficients are zero, so $\text{Null}(A_d) = \{0\}$ and A_d has full column rank.

Conversely, suppose the distinct observed inputs are $s_1, \dots, s_k$, with $k \leq d$. Consider

$$p(t) = \prod_{j=1}^k (t - s_j). \quad (4.3)$$

This is a nonzero polynomial of degree $k \leq d$ that vanishes at every observed input, including repeated observations. Its coefficient vector, padded with zeros through degree d if necessary, is therefore a nonzero vector in $\text{Null}(A_d)$. Hence A_d does not have full column rank. Repeated measurements do not supply additional distinct roots, which is why the number of distinct inputs determines this rank condition. ■

Example 6 (A quadratic fit that reveals curvature). Consider the five observations

t_i	-2	-1	0	1	2
b_i	3.1	0.6	1.6	2.6	7.1

For a line, $\bar{t} = 0$, $\bar{b} = 3$, $\sum_i t_i^2 = 10$, and $\sum_i t_i b_i = 10$. Hence

$$\hat{f}_1(t) = 3 + t, \quad \mathbf{r}_1 = (2.1, -1.4, -1.4, -1.4, 2.1)^\top, \quad \text{SSE}_1 = 14.7. \quad (4.4)$$

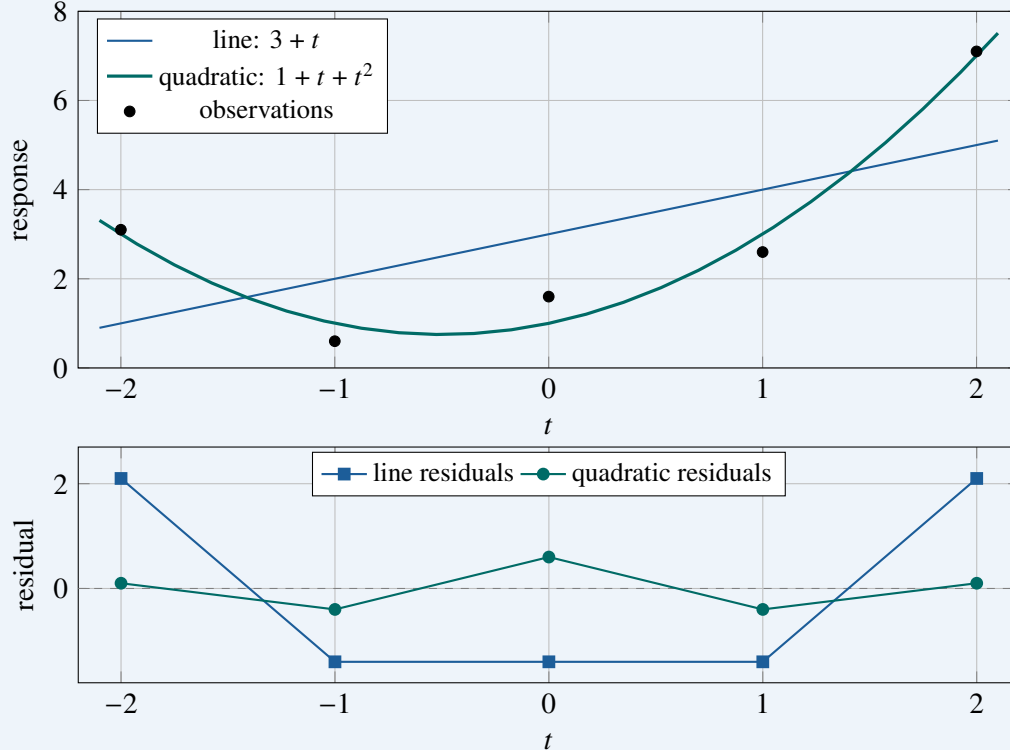
For a quadratic, the normal equations are

$$\begin{pmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 15 \\ 10 \\ 44 \end{pmatrix}. \quad (4.5)$$

The middle equation gives $c_1 = 1$. Subtracting twice the first from the third gives $14c_2 = 14$, and then $c_0 = 1$. Therefore

$$\hat{f}_2(t) = 1 + t + t^2, \quad \mathbf{r}_2 = (0.1, -0.4, 0.6, -0.4, 0.1)^\top, \quad \text{SSE}_2 = 0.7. \quad (4.6)$$

Verify $\sum_i r_i = \sum_i t_i r_i = \sum_i t_i^2 r_i = 0$. In particular, $4(0.1) - 0.4 - 0.4 + 4(0.1) = 0$. The residual is perpendicular to all three columns of A_2 .



The line's residuals are positive at both ends and negative in the middle: a curved pattern remains after fitting. The quadratic removes that broad pattern. Lines joining residual markers only guide the eye; there are no additional measurements between markers. Five points alone are too few to establish that the remaining errors are random.

5 Choosing between a line and a higher-degree polynomial

The best degree depends on the signal, noise, number and placement of inputs, and intended use. An underlying scientific relationship may already favor a line or quadratic. Otherwise, compare a small set of plausible degrees using residuals and prediction on data that were not used to fit the coefficients [4, Chs. 5–7].

5.1 Why training error alone cannot choose the degree

Theorem 4 (Adding polynomial terms cannot increase the best training fit error). Let SSE_d denote the minimum squared residual for polynomials of degree at most d on a fixed dataset. Then

$$\text{SSE}_{d+1} \leq \text{SSE}_d. \quad (5.1)$$

Proof. Every candidate of degree at most d is also a candidate of degree at most $d + 1$: set its final coefficient to zero. Minimizing over the larger set cannot give a larger minimum. ■

This statement is about exact, unregularized least squares. Finite precision or numerical rank truncation can affect a computed comparison. With m distinct inputs, degree $m - 1$ can interpolate the data exactly, including measurement noise.

Definition 3 (Underfitting and overfitting). *Underfitting* occurs when the model is too restricted to describe the relevant relationship. *Overfitting* occurs when the fit follows features of the training data, including noise, that do not predict new responses well.

A line fitted to a strongly curved relationship may underfit. A high-degree polynomial through noisy observations may overfit. A visually smooth curve can still overfit, and a higher degree is not automatically wrong; prediction must be checked.

5.2 Measure performance on held-out data

A regression fit is a function, so it can be evaluated at inputs that were absent from the training data. Let $\phi(t) = (\phi_0(t), \dots, \phi_{p-1}(t))^T$ collect the chosen basis functions. After learning $\mathbf{x}_* = A^+ \mathbf{b}$, predict at a new input t_* by

$$\widehat{b}_* = \widehat{f}(t_*) = \phi(t_*)^T \mathbf{x}_* = \phi(t_*)^T A^+ \mathbf{b}. \quad (5.2)$$

The unknown response b_* is not needed to make this prediction. Keep the fitted coefficients, basis functions, and any training-derived scaling fixed. For several new inputs, construct a new design matrix A_{new} with the *same columns of features* and compute $\widehat{\mathbf{b}}_{\text{new}} = A_{\text{new}} \mathbf{x}_*$. Evaluating a new input does not require refitting.

Adding that observation to the supervised training set requires its response; a later refit may also change the basis or penalty using data already available.

Divide the available observations into a training set T and a validation set V before fitting. For each candidate degree d , fit only on T , obtaining $\hat{f}_{d,T}$. Then calculate

$$\text{MSE}_{\text{train}}(d) = \frac{1}{|T|} \sum_{i \in T} [b_i - \hat{f}_{d,T}(t_i)]^2, \quad \text{MSE}_{\text{val}}(d) = \frac{1}{|V|} \sum_{i \in V} [b_i - \hat{f}_{d,T}(t_i)]^2. \quad (5.3)$$

Use the same split and evaluation measure for every degree. Dividing by the number of observations makes training and validation errors comparable in scale.

5.3 Two datasets, two different degree choices

With five distinct inputs, a polynomial of degree at most four can interpolate all five responses. For Example 6, it is

$$\hat{f}_4(t) = \frac{8}{5} + t - \frac{7}{24}t^2 + \frac{7}{24}t^4, \quad \text{SSE}_4 = 0. \quad (5.4)$$

Interpolation enforces agreement at the observed inputs. Regression permits disagreement to obtain a simpler or more useful description of noisy data. A zero training residual is not evidence of accurate prediction between observations or beyond the observed range.

Example 7 (Choosing the degree by checking predictions). We will compare polynomials of degrees at most 1, 2, and 4 on two small datasets. The question is not simply which polynomial passes closest to the observations used to fit it, but which one predicts *other* observations most accurately. Each candidate is fitted by least squares; the validation data then help us choose between the candidates.

Step 1: Separate fitting from checking. For each dataset, use the five training observations below to determine the coefficients. Reserve the four validation observations to check the resulting predictions. The two response rows describe *separate* problems with the same input values; do not combine their responses.

Training inputs t	-2	-1	0	1	2
Curved data b	3.1	0.6	1.6	2.6	7.1
Nearly linear data b	-0.7	-0.5	1.4	1.5	3.3
Validation inputs t	-1.5	-0.5	0.5	1.5	
Curved data b	1.80	0.70	1.70	4.80	
Nearly linear data b	-0.45	0.45	1.45	2.55	

These are constructed teaching datasets: the curved responses were obtained by adding small errors to $1 + t + t^2$, and the nearly linear responses by adding small errors to $1 + t$. Both validation sets add the errors $(0.05, -0.05, -0.05, 0.05)$ to their respective signals. We know these underlying functions because we constructed the example; with real measurements, they would generally be unknown. The fitting procedure uses only the observed training pairs, not this knowledge of how the data were generated.

Step 2: Determine each candidate from the training data only. For degree d , form the training design matrix whose i th row is $(1, t_i, \dots, t_i^d)$. With $\mathbf{b}_T$ denoting one row of training responses written as a

column, compute

$$\mathbf{x}_{d,T} = A_{d,T}^+ \mathbf{b}_T, \quad \hat{f}_d(t) = (1, t, \dots, t^d) \mathbf{x}_{d,T}. \quad (5.5)$$

There are $d + 1$ coefficients: two for a line, three for a quadratic, and five for a quartic. The five distinct training inputs give full column rank for each of these degrees, so each least-squares coefficient vector is unique. No validation response enters this calculation.

Here is how to reproduce the line and quadratic coefficients by hand. For the training inputs, symmetry gives

$$\sum_{i=1}^5 t_i = \sum_{i=1}^5 t_i^3 = 0, \quad \sum_{i=1}^5 t_i^2 = 10, \quad \sum_{i=1}^5 t_i^4 = 34. \quad (5.6)$$

Consequently, the normal equations for a line $c_0 + c_1 t$ reduce to

$$5c_0 = \sum_{i=1}^5 b_i, \quad 10c_1 = \sum_{i=1}^5 t_i b_i. \quad (5.7)$$

The curved data have $\sum_i b_i = 15$ and $\sum_i t_i b_i = 10$, giving $\hat{f}_1(t) = 3 + t$. The nearly linear data have $\sum_i b_i = 5$ and $\sum_i t_i b_i = 10$, giving $\hat{f}_1(t) = 1 + t$.

For a quadratic $c_0 + c_1 t + c_2 t^2$, the normal equations become

$$\begin{pmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^5 b_i \\ \sum_{i=1}^5 t_i b_i \\ \sum_{i=1}^5 t_i^2 b_i \end{pmatrix}. \quad (5.8)$$

The right-hand side is $(15, 10, 44)^\top$ for the curved data and $(5, 10, 57/5)^\top$ for the nearly linear data. The middle equation gives $c_1 = 1$ in both cases. Subtracting twice the first equation from the third eliminates c_0 , so

$$c_2 = \frac{\sum_i t_i^2 b_i - 2 \sum_i b_i}{14}, \quad c_0 = \frac{\sum_i b_i - 10c_2}{5}. \quad (5.9)$$

Thus the curved data give $(c_0, c_1, c_2) = (1, 1, 1)$, as in Example 6, whereas the nearly linear data give $(c_0, c_1, c_2) = (4/5, 1, 1/10)$.

For degree 4, the design matrix is square and invertible, so solving $A_{4,T} \mathbf{x}_{4,T} = \mathbf{b}_T$ gives a polynomial that passes through every training point. The curved-data polynomial is already given in (5.4); the nearly linear one appears below. Substitution of the five training inputs verifies both interpolants. Their zero training errors follow from having enough coefficients to interpolate these five distinct inputs, not from knowing the underlying relationship.

Step 3: Check the curved-data fits at new inputs. Keep each fitted polynomial fixed and substitute the four validation inputs. For example, at the previously unused input $t = -1.5$, the quadratic predicts

$$\hat{f}_2(-1.5) = 1 - 1.5 + (-1.5)^2 = 1.75. \quad (5.10)$$

The observed validation response is 1.80, so the residual (observed minus predicted) is $1.80 - 1.75 = 0.05$ and its squared value is 0.0025. The response 1.80 is needed to *check* the prediction, but not to *make* it. Repeating the substitution gives the following table; the quartic predictions are rounded here to six decimal places.

Input t	Observed b	Line $\hat{f}_1(t)$	Quadratic $\hat{f}_2(t)$	Quartic $\hat{f}_4(t)$
-1.5	1.80	1.50	1.75	0.920313
-0.5	0.70	2.50	0.75	1.045313
0.5	1.70	3.50	1.75	2.045313
1.5	4.80	4.50	4.75	3.920313

For each column of predictions, square the four residuals, add them, and divide by 4. In particular,

$$\text{MSE}_{\text{val}}(1) = \frac{(0.30)^2 + (-1.80)^2 + (-1.80)^2 + (0.30)^2}{4} = 1.665, \quad (5.11)$$

$$\text{MSE}_{\text{val}}(2) = \frac{(0.05)^2 + (-0.05)^2 + (-0.05)^2 + (0.05)^2}{4} = 0.0025. \quad (5.12)$$

For the quartic, the exact validation residuals are $(563, -221, -221, 563)^T/640$, giving

$$\text{MSE}_{\text{val}}(4) = \frac{2(563/640)^2 + 2(221/640)^2}{4} = \frac{36581}{81920} \approx 0.446545. \quad (5.13)$$

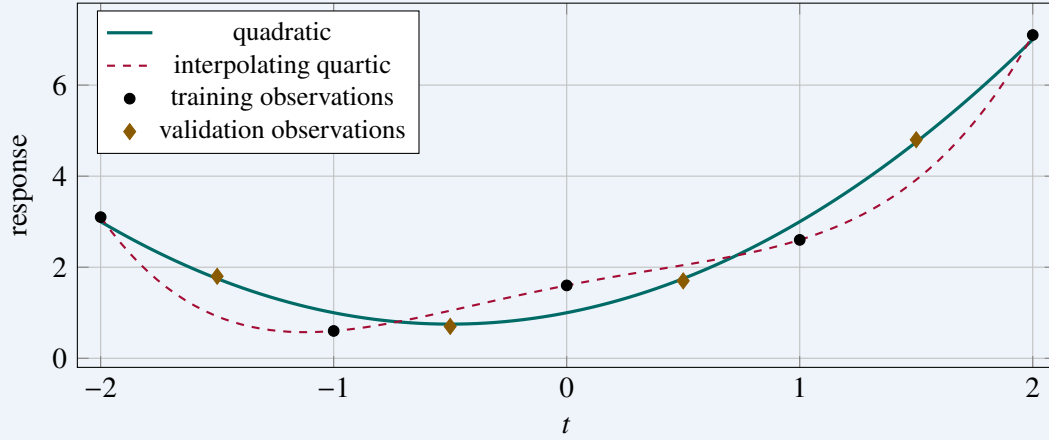
Training MSE uses the same rule, but averages over the *five training points*. For instance, the quadratic's training residuals from Example 6 give

$$\text{MSE}_{\text{train}}(2) = \frac{(0.1)^2 + (-0.4)^2 + (0.6)^2 + (-0.4)^2 + (0.1)^2}{5} = 0.140. \quad (5.14)$$

Here MSE is the average squared prediction error, so we divide by the number of observations, not by that number minus the number of coefficients. Collecting the results gives

d	Fitted function	Training MSE	Validation MSE
1	$3 + t$	2.940	1.665000
2	$1 + t + t^2$	0.140	0.002500
4	$\frac{8}{5} + t - \frac{7}{24}t^2 + \frac{7}{24}t^4$	0	0.446545

Decision for the curved data: choose the quadratic among these candidates. The line misses the broad curvature and underfits. The quartic matches all five noisy training responses exactly, but its extra flexibility worsens predictions at the four held-out inputs compared with the quadratic: this illustrates overfitting. Notice that the quartic still predicts better than the line here; the relevant comparison is between all three validation errors.



This plot shows the curved dataset. The quartic reaches every black training marker, but the quadratic passes closer to the gold validation markers. A perfect fit at the black markers does not force accurate predictions between them.

Step 4: Repeat the comparison for the nearly linear data. Using its training responses, the same fitting and scoring procedure gives

d	Fitted function	Training MSE	Validation MSE
1	$1 + t$	0.168	0.002500
2	$\frac{4}{5} + t + \frac{1}{10}t^2$	0.140	0.008125
4	$\frac{7}{5} + t - \frac{143}{120}t^2 + \frac{7}{24}t^4$	0	0.379749

To check the first two validation scores, the line predicts $(-0.5, 0.5, 1.5, 2.5)$ and the quadratic predicts $(-0.475, 0.325, 1.325, 2.525)$, in increasing order of the validation inputs. Subtracting these predictions from the observed responses gives

$$\text{MSE}_{\text{val}}(1) = \frac{(0.05)^2 + (-0.05)^2 + (-0.05)^2 + (0.05)^2}{4} = 0.0025, \quad (5.15)$$

$$\text{MSE}_{\text{val}}(2) = \frac{(0.025)^2 + (0.125)^2 + (0.125)^2 + (0.025)^2}{4} = 0.008125. \quad (5.16)$$

Decision for the nearly linear data: choose the line among these candidates. The quadratic reduces training MSE from 0.168 to 0.140, but the added curvature increases validation MSE. The quartic again attains zero training error and performs much worse on validation. A more flexible model is able to follow more of the training variation; whether that variation is useful for prediction is a separate question.

What this comparison establishes. There are two decisions: least squares chooses the *coefficients* for a *fixed degree*, while validation helps choose the *degree itself*. Always compare the candidates on the same held-out observations, without refitting their coefficients to those observations before scoring them.

Here the validation scores favor a quadratic for one dataset and a line for the other. This does not establish a universal best degree, and four validation observations are too few to justify a reliable conclusion in an application. The validation responses also happen to have smaller errors than the training responses,

which explains why the selected model's validation MSE is smaller than its training MSE; there is no rule that training MSE must always be smaller.

All four validation inputs lie inside the training range $[-2, 2]$. These checks assess prediction at unseen inputs *within* that range, not extrapolation beyond it. Once validation has been used to select the degree, an independent test set is needed for a separate assessment of the selected model's predictive performance. No test set is provided in this small example.

6 From least squares to neural networks

We first compute a fit when the features are fixed, then interpret the same task as supervised learning. Finally, we ask what changes when a model learns its features as well as its coefficients. The distinction determines whether one linear solve suffices or a more general optimization problem is required.

6.1 Least squares as supervised learning

In *supervised learning*, each training observation contains an input and an observed target response. A learning procedure uses these pairs to construct a rule for predicting responses to future inputs. Regression is the case in which the target is numerical. Thus the line and curve fits in this note are simple supervised-learning models; “training” means estimating their coefficients from the supplied observations [4, Chs. 2–3].

Definition 4 (Features, model, and training loss). For observations (t_i, b_i) , choose a vector of known features

$$\boldsymbol{\phi}(t) = (\phi_0(t), \dots, \phi_{p-1}(t))^T, \quad f_{\mathbf{x}}(t) = \boldsymbol{\phi}(t)^T \mathbf{x}. \quad (6.1)$$

The i th row of the design matrix is $\boldsymbol{\phi}(t_i)^T$. With squared loss $\ell(b, \hat{b}) = (b - \hat{b})^2$, the average training loss, or *empirical risk*, is

$$\widehat{R}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^m \ell(b_i, f_{\mathbf{x}}(t_i)) = \frac{1}{m} \|\mathbf{b} - A\mathbf{x}\|_2^2. \quad (6.2)$$

For a line, the features are $(1, t)$; for a quadratic, they are $(1, t, t^2)$. An input can also contain several measurements, with features such as a constant, temperature, and pressure. The same matrix construction applies. Features must be available when making the intended prediction; the unknown future response cannot be used as an input.

Minimizing (6.2) is exactly the least-squares problem already studied. The pseudoinverse therefore gives the training rule

$$\mathbf{x}_* = A^+ \mathbf{b}, \quad \widehat{f}(t_*) = \boldsymbol{\phi}(t_*)^T A^+ \mathbf{b}. \quad (6.3)$$

The first expression learns coefficients; the second evaluates the learned rule. If the design is rank deficient, the pseudoinverse chooses the minimum-norm coefficients in the specified coordinates. Training responses alone may then leave predictions at new inputs undetermined [3].

Example 8 (Training a line and predicting a new response). For the Lecture 5 pairs $(0, 1), (1, 2), (2, 2)$, use $\phi(t) = (1, t)^\top$. Training gives $\mathbf{x}_* = (7/6, 1/2)^\top$, with SSE $1/6$ and empirical risk $1/18$. At the new input $t_* = 3/2$, the prediction is

$$\hat{f}(3/2) = \begin{pmatrix} 1 & 3/2 \end{pmatrix} \begin{pmatrix} 7/6 \\ 1/2 \end{pmatrix} = \frac{23}{12}. \quad (6.4)$$

If a response $b_* = 2$ is subsequently observed there, its squared prediction error is $(2 - 23/12)^2 = 1/144$. This response was not needed to make the prediction. One such error illustrates evaluation; it is too little evidence to establish predictive reliability.

6.2 From fixed features to learned features

What changes when we replace polynomial regression by a neural network? We can keep the same input–response pairs (t_i, b_i) , the same prediction task, and even the same squared-error loss. The important change is in the *family of functions used to model the response*.

For degree- d polynomial regression, the features $1, t, \dots, t^d$ are chosen before coefficient fitting. The unknowns only specify how much of each fixed feature to add. The function can curve as a function of t , but it is linear in its coefficient vector. The least-squares *objective* is quadratic, not linear, in that vector.

A neural network can learn the features as well as their final combination. For one scalar input and one hidden layer with q units, a regression model is

$$f_\theta(t) = c + \sum_{j=1}^q a_j \sigma(w_j t + v_j), \quad \theta = (c, a_1, \dots, a_q, w_1, \dots, w_q, v_1, \dots, v_q)^\top. \quad (6.5)$$

Here w_j is an input weight, v_j is a hidden-unit bias, a_j is an output weight, and c is an output bias. The activation σ is a chosen function, such as $\tanh$ or the rectified linear unit $\sigma(s) = \max(0, s)$, called ReLU. The quantities $\sigma(w_j t + v_j)$ are the learned features. For a vector input $\mathbf{t}$, replace $w_j t$ by an inner product $\mathbf{w}_j^\top \mathbf{t}$. A deeper network composes several such layers, learning transformations of earlier features [6, Ch. 6].

Freezing the hidden layer recovers least squares. If all w_j and v_j are fixed, form the known matrix

$$\Phi_{i,:} = (1, \sigma(w_1 t_i + v_1), \dots, \sigma(w_q t_i + v_q)), \quad \boldsymbol{\beta} = (c, a_1, \dots, a_q)^\top. \quad (6.6)$$

With a linear output and squared loss, the minimum-norm output-layer fit is exactly $\boldsymbol{\beta}_* = \Phi^+ \mathbf{b}$. Nonlinear features alone therefore do not prevent a linear least-squares solve. When the hidden weights are trained too, Φ changes with those weights; there is no single fixed design matrix whose pseudoinverse determines all network parameters. A nonlinear output activation or a different loss can also change the output-layer optimization problem.