

Linear Regression

Daniel Enriquez Recitation 1 Fall 2026

1 Learning from examples

Imagine taking the same MTA route to class each morning. At first, you leave with plenty of time. After a few weeks, you have a sense of when to head out and how much room to leave for the transfer. Then one morning the train stops between stations. The minutes pass. This was not the delay you had allowed for. The surprise tells us something: from the trips you had already made, you had formed an expectation about a new one.

Suppose you start keeping a record. For each journey, you note the day, departure time, any service alert available when you leave, and the time you reach your destination. This collection of observations is a *dataset*. Each row describes one trip, and its recorded duration has a definite meaning: perhaps it runs from entering the first station to exiting the last, including the wait and transfer. We have preserved a few aspects of the journey, enough to compare one morning with another, while leaving much of what happened unrecorded.

Now consider tomorrow's trip. Its duration is still unknown, but the route, departure time, and posted alerts are available. These are possible *features*; the duration we want to estimate is the *target*. A *hypothesis function* takes the features as input and returns an estimate of the target; that output is our *prediction*. We can judge these predictions on completed trips by comparing them with the durations we recorded. Learning such a function from input–target examples is called *supervised learning*.

The task is more interesting than remembering the table. A function that reproduces every recorded duration may have nothing useful to say about tomorrow. We want *generalization*: a pattern learned from earlier trips that remains useful on new ones. Even a useful hypothesis function will sometimes miss a delay. More records may improve our understanding of the usual pattern, while an individual trip still varies for reasons our features do not reveal. Recording different information can help with that second difficulty in ways that simply recording more trips cannot.

This is a small version of an extraordinary ability. We routinely let a handful of experiences change how we act in situations we have never encountered. Machine learning asks how to make that process explicit: what to represent, what to fit, and what evidence would justify trusting the result. The subway example gives those questions a practical meaning. We care about the learned function because it will be used after the last row of the dataset, when the answer is no longer there to check.

But which function should we choose? Many functions can agree on the recorded examples and disagree on the next case. The data alone do not settle how a pattern should extend beyond them. We supply some structure by choosing a *hypothesis class* $\mathcal{H}$: the set of hypothesis functions we are willing to consider. Fitting means selecting a function from this class using the observations. We might allow only a constant travel time, or let the estimate vary with departure time. The second class can express relationships the first cannot. Neither includes every imaginable function.

An *inductive bias* is the set of assumptions and preferences a learning method uses to favor some hypothesis functions over others, beyond what the training data alone justify. Restricting the hypothesis class rules out some possibilities entirely; preferring smoother functions within a class favors some without excluding the rest. For our commute, a smoothness preference would make nearby departure times tend to receive similar predictions, even when one time has never been

observed. That can be useful, but it may miss an abrupt change in service. The question is whether the bias suits the problem. Linear regression gives us a concrete commitment to examine. To make the choice of a hypothesis class concrete, imagine a small experiment: grow seedlings under different amounts of daily light and measure their heights after two weeks. We would like to estimate the height of a new plant from its growing conditions. Predicting a numerical target like height is a regression task. Each observation pairs an input $x^{(i)}$ with a target $y^{(i)}$, giving the training dataset

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N.$$

Here N is the number of observations, also called training examples; parenthesized superscripts index them. With light as our only feature, $x^{(i)}$ is a single number: the hours of daily light for plant i . We write h for a hypothesis function and $h(x)$ for its prediction at input x : here, an estimated height. During training we know both inputs and targets; for a new plant we use its available input to predict the target we have not measured.

The measurements in Figure 1 are scattered, as we would expect when light is only part of the explanation for height. A straight line follows the broad upward trend. A quadratic, a cubic, and a curve that approaches a plateau can each follow the observations reasonably well too. None passes through every point. Within the measured range, their differences are modest beside the scatter; beyond it, their predictions separate dramatically. This sample of observations offers some evidence for comparing the functions, but it has not told us whether the trend should keep rising, level off, or eventually turn downward.

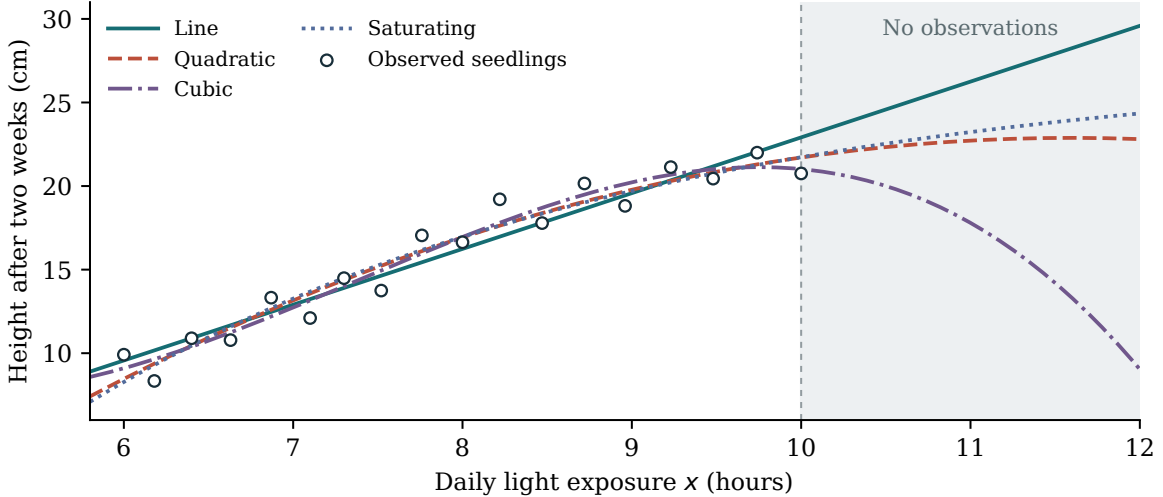


Figure 1: Eighteen synthetic seedling measurements and four candidate fits from different function families. Each captures the observed trend with some error. Their predictions diverge in the shaded region, where no seedlings were observed. The curves are fitted candidates, not known biological laws.

There is an even sharper way to see what finite observations leave unresolved. Suppose light is the only feature and its observed values are distinct. Start with any hypothesis function h , whether or not it passes through the data. Multiplying together the differences between a proposed input z and every observed input gives a term that vanishes at each observed value. We can therefore construct

$$h_c(z) = h(z) + c \prod_{i=1}^N (z - x^{(i)}), \quad c \in \mathbb{R}. \quad (1)$$

Every h_c makes exactly the same training predictions as h , and inherits whatever errors h already makes. At an unobserved input the product is nonzero, so varying c changes the prediction by any amount. Unlike the four fitted curves in the figure, these constructed functions have identical training errors. Even that stronger agreement would not determine what happens at a new input.

Choosing a class $\mathcal{H}$ gives the examples a way to constrain one another. If we allow only straight lines, moving a prediction at one input changes predictions elsewhere as well. We have tied them together through a shared slope and intercept. This is why restricting the class can help us learn: an observed relationship has consequences beyond the particular points at which it was observed. The restriction is not merely a convenience for the computer.

It is also a commitment we can get wrong. A straight line cannot describe a growth pattern that rises and then levels off. A more flexible class can capture that shape, but also admits more ways to follow accidental fluctuations in a small sample. The aim is to find assumptions that preserve useful structure while ruling out unsupported freedom. Restricting $\mathcal{H}$ is one way to express that preference; later, *regularization* will let us prefer some functions within a larger class. Neither choice guarantees generalization. It makes explicit what we are asking the data to choose among.

2 Linear regression

Linear regression restricts its hypothesis class to *linear models*. Suppose an input contains d recorded features $x_1, \dots, x_d$. For a plant, light exposure and water supply might both help explain its height. A linear model assigns a coefficient to each feature and adds their contributions to an intercept. To include the intercept in the same vector calculation, prepend a constant feature $x_0 = 1$ and collect the inputs and coefficients as

$$x = \begin{bmatrix} 1 & x_1 & \cdots & x_d \end{bmatrix}^T, \quad w = \begin{bmatrix} w_0 & w_1 & \cdots & w_d \end{bmatrix}^T, \quad x, w \in \mathbb{R}^{d+1}.$$

Here T denotes transpose. The *dot product* turns these two column vectors into a prediction:

$$h_w(x) = w^T x = \sum_{j=0}^d w_j x_j = w_0 + w_1 x_1 + \cdots + w_d x_d. \quad (2)$$

The transpose turns w into a row; multiplying it by x pairs the matching coordinates and adds their products. The coefficients are the model's *parameters*. The *intercept* w_0 sets the prediction when every recorded feature is zero, while the constant feature remains one. Each *slope* w_j specifies how much the prediction changes when feature j increases by one and the others stay fixed. Allowing all coefficient vectors gives the hypothesis class

$$\mathcal{H}_{\text{lin}} = \{h_w : x \mapsto w^T x \mid w \in \mathbb{R}^{d+1}\}, \quad x_0 = 1. \quad (3)$$

Every input here includes that leading one; there are still only d recorded features. Each parameter vector specifies one hypothesis function in this class. Fitting means selecting its coefficients using the observations, which need not agree exactly with its predictions. At observation i , the prediction is $\hat{y}^{(i)} = h_w(x^{(i)}) = w^T x^{(i)}$, with a hat to distinguish it from the observed target.

The graph of this function is a *hyperplane* in $\mathbb{R}^{d+1}$, whose coordinates are the d recorded features and the predicted target. A hyperplane is a flat set described by one linear equation. With the constant feature $x_0 = 1$ understood, we can write that equation in either form:

$$\hat{y} = w^T x \quad \Longleftrightarrow \quad \hat{y} - \sum_{j=1}^d w_j x_j = w_0.$$

With two recorded features the graph is a plane in three-dimensional space; with one it is a line in the plane. The slope of each feature is constant wherever we are on this surface. That is a statement about the model; an observational relationship alone does not establish the effect of intervening in the world.

For example, with light as the only recorded feature, $w^\top x = w_0 + w_1 x_1$: each extra hour adds w_1 to the predicted height. Here the intercept extrapolates to zero light, outside our observed range of six to ten hours, so it need not have a useful biological interpretation.

Strictly, a function with a nonzero intercept is *affine*; a linear function must pass through the origin. The conventional name *linear regression* allows an intercept and refers to linear dependence on the coefficients. The model assumes additive contributions with constant slopes. A curved relationship, or a slope that depends on another feature, requires a richer representation. First we need a way to choose the best fit within this class; its errors can then help us judge what the class misses.

Choosing a hypothesis class tells us which functions are available, but it does not tell us which one to prefer. We need to decide what a poor prediction costs. A *loss function* $\mathcal{L}(y, \hat{y})$ assigns a numerical penalty to a prediction $\hat{y}$ when the observed target is y ; smaller loss means a better prediction by that criterion. Absolute error, $|y - \hat{y}|$, and squared error, $(y - \hat{y})^2$, are two possible choices. They express different preferences about how much a large error should matter. The hypothesis class and the loss are separate modeling decisions: the same class can be fitted using either loss.

One prediction is not enough to compare two functions. A candidate may do better on one plant and worse on another. Giving each training observation equal weight, we summarize its performance by averaging the losses over the dataset. This average is the *empirical risk* of h :

$$\hat{R}_{\mathcal{D}}(h) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^{(i)}, h(x^{(i)})). \quad (4)$$

The word empirical means that the average uses the observations we have, rather than all the cases on which we might eventually predict. To compare functions using this average, we keep the dataset and loss fixed. Fitting by *empirical risk minimization* means seeking a member of $\mathcal{H}$ with the smallest such average. When the minimum is attained, a fitted hypothesis function $\hat{h}$ satisfies

$$\hat{h} \in \arg \min_{h \in \mathcal{H}} \hat{R}_{\mathcal{D}}(h). \quad (5)$$

Here $\arg \min$ denotes the set of minimizers, and $\in$ allows more than one function to attain the same minimum. This makes our task precise, but it is still a task about the training data. A small empirical risk does not by itself establish good predictions on new observations.

An *objective function* is the quantity we ask the fitting procedure to optimize. Empirical risk serves as the objective in this formulation; other objectives can combine it with a penalty such as regularization. When we report performance, we use a numerical summary called a *score*. To interpret that score, we must specify its direction: smaller mean squared error is better, whereas larger R^2 , introduced later, is better. A score may also serve as an objective, but reporting a score does not mean we optimized it. These names describe the role a quantity plays, not three interchangeable names for prediction error.

For least-squares linear regression, we choose the squared loss $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$. The signed discrepancy at observation i is the *residual*,

$$\epsilon^{(i)}(w) = y^{(i)} - h_w(x^{(i)}).$$

A positive residual means that we predicted too little. Simply adding residuals would allow overprediction on one plant to cancel underprediction on another. Squaring makes each contribution nonnegative, and a residual twice as large contributes four times as much loss. This gives large discrepancies considerable influence. It is a preference we have chosen, not a consequence of using a linear hypothesis class.

On a plot of light exposure and height, the residuals are vertical: they compare the predicted and observed heights at the supplied input. Perpendicular distance to the line would combine input and target deviations and would define a different fitting problem. With squared loss, the total training loss is called the *residual sum of squares*:

$$\text{RSS}(w) = \sum_{i=1}^N \mathcal{L}(y^{(i)}, h_w(x^{(i)})) = \sum_{i=1}^N (y^{(i)} - h_w(x^{(i)}))^2. \quad (6)$$

Its average is exactly the empirical risk. For this loss, the same average is called the mean squared error, or the in-sample error E_{in} when evaluated on the training data:

$$\hat{R}_{\mathcal{D}}(h_w) = \frac{\text{RSS}(w)}{N} = \text{MSE}(w) = E_{\text{in}}(w). \quad (7)$$

Averaging keeps this quantity unchanged if the dataset is duplicated. For the calculations that follow, we include one further constant factor and use the training objective

$$J(w) = \frac{1}{2} \hat{R}_{\mathcal{D}}(h_w) = \frac{1}{2N} \text{RSS}(w). \quad (8)$$

The factor $1/2$ will cancel the factor of two when a square is differentiated. Multiplication by a positive constant changes the numerical value of the objective but not its minimizing parameters. Thus minimizing J , empirical risk, or RSS gives the same least-squares fit. We use w^{lin} for a minimizing parameter vector; the superscript *lin* distinguishes it from an arbitrary candidate w . We have now specified both the functions we can choose and how to compare them. What remains is to find out what the best choice can explain.

3 What the best fit can explain

We can write down the empirical risk of any candidate, but finding its minimum requires understanding how the predictions can change together. With several features, their observed relationships matter: two measurements of size, for example, may carry overlapping information. We need a way to describe all the training predictions obtainable from our hypothesis class. A matrix provides this description. Each augmented input $x^{(i)}$ already includes the constant feature $x_0^{(i)} = 1$. Stacking the row vectors $(x^{(i)})^T$ gives the design matrix

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times (d+1)}, \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} \in \mathbb{R}^N. \quad (9)$$

The vector Xw contains all N training predictions. To see how, read a row of X as one example and a column as one feature recorded across the examples. Multiplying each column by its coefficient

gives that feature's contribution to every prediction at once. Adding those contributions, including the constant column, produces Xw . Nothing about the hypothesis class has changed; we have written the same hypothesis function in a form that lets us compare whole sets of predictions.

The residual vector $y - Xw$ contains the residuals we decided to square and add. We can express that sum using vector length: for a vector a of N real coordinates, $a^\top a = \sum_i a_i^2$, and its Euclidean norm is $\|a\|_2 = \sqrt{a^\top a}$. The dot product used here, also called the *inner product* for real vectors, has the form $a^\top b = \sum_i a_i b_i$ and measures how two vectors align. Our chosen objective is therefore also a squared distance between observed and predicted target vectors:

$$J(w) = \frac{1}{2N} \|y - Xw\|_2^2, \quad w^{\text{lin}} \in \arg \min_{w \in \mathbb{R}^{d+1}} J(w). \quad (10)$$

This is the same minimization problem expressed in matrix form. Membership in the set of minimizers leaves room for more than one parameter vector to fit equally well. The norm in this expression is taken in $\mathbb{R}^N$, where each coordinate corresponds to an observation. The distance it measures is the distance between two vectors of target values, rather than between two parameter vectors.

We can now ask which target patterns the model is capable of reproducing on these observations. Varying w forms every possible linear combination of the columns of X . The set of those vectors is called the column space:

$$\mathcal{S} = \text{col}(X) = \{Xw : w \in \mathbb{R}^{d+1}\}.$$

This column space is closed under addition and scalar multiplication, and hence is a subspace of $\mathbb{R}^N$. This is the geometric consequence of our hypothesis class: we can choose only prediction vectors in this space. Within this space, the fitting problem asks for the point closest to y . There need not be a vector w satisfying $Xw = y$: a subspace spanned by a few feature columns ordinarily cannot contain every possible vector of N target values.

Imagine a candidate prediction vector and the residual that remains between it and y . If part of the residual points along a direction our feature columns can produce, we should be able to move the fit a little in that direction and reduce the error. At the best fit, no such component should remain. Orthogonality makes this intuition precise: two vectors are orthogonal when their inner product is zero. Expanding a squared distance gives the identity

$$\|a + b\|_2^2 = \|a\|_2^2 + 2a^\top b + \|b\|_2^2.$$

When a and b are orthogonal, the cross term vanishes and this becomes the Pythagorean theorem. Suppose $p \in \mathcal{S}$ and $y - p$ is orthogonal to every vector in $\mathcal{S}$. For any $q \in \mathcal{S}$, the difference $p - q$ also belongs to $\mathcal{S}$, so

$$\|y - q\|_2^2 = \|(y - p) + (p - q)\|_2^2 = \|y - p\|_2^2 + \|p - q\|_2^2. \quad (11)$$

This is a satisfying place to stop and read the equation. Once the residual is perpendicular to the space of allowed predictions, moving the fit anywhere else in that space can only add the squared length $\|p - q\|_2^2$ to the error. There is no remaining direction in which to improve it. The right side is smallest at $q = p$, so the closest prediction vector is unique.

Such a point exists for every finite-dimensional subspace. We can construct it by choosing an orthonormal basis $u_1, \dots, u_r$ for $\mathcal{S}$: mutually perpendicular unit vectors spanning the same space as the feature columns. One way to obtain them is to subtract from each independent column its components along the directions already chosen, then normalize the remainder. Along a unit

direction u_k , the inner product $u_k^\top y$ measures how much of y points that way. Retaining all the components the model can express gives

$$p = \sum_{k=1}^r (u_k^\top y) u_k.$$

This vector lies in $\mathcal{S}$, and for each basis vector, $u_k^\top (y - p) = 0$, so the remainder is perpendicular to the entire subspace. The Pythagorean identity therefore proves that p is the unique closest point. This is the orthogonal projection of y onto $\mathcal{S}$.

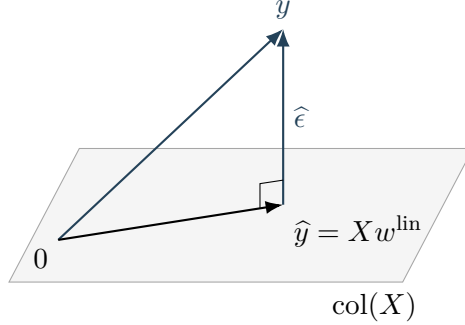


Figure 2: The geometry is in $\mathbb{R}^N$, the space of training prediction vectors. The plane represents the column space schematically. The fitted residual is orthogonal to every direction the model can express.

Because $p \in \text{col}(X)$, at least one coefficient vector produces it. Let w^{lin} be such a vector, write $\hat{y} = Xw^{\text{lin}}$, and denote the fitted residual by $\hat{\epsilon} = y - Xw^{\text{lin}}$. Orthogonality to every column is the matrix equation $X^\top \hat{\epsilon} = 0$, or

$$X^\top X w^{\text{lin}} = X^\top y. \quad (12)$$

These are the normal equations; the word normal refers to perpendicularity. They can look like an algebraic recipe if we meet them only as a formula to solve. Here they say exactly what we hoped a good fit would do: leave no residual component that another combination of the recorded features could explain. Since the intercept column is $\mathbf{1}$, one of the equations is $\mathbf{1}^\top \hat{\epsilon} = 0$: the residuals sum to zero, and the fitted predictions have the same mean as the observed targets.

Conversely, if a candidate residual has a nonzero inner product with some feature direction, a small movement along that direction can reduce its squared length. Every minimizing fit must therefore satisfy the normal equations. We have both a solution and a way to recognize one: its residual is orthogonal to all the available feature columns. That is a strong statement, but a deliberately limited one. A pattern outside their span can remain completely unexplained.

There is still a possible ambiguity. A unique fitted prediction vector need not come from unique coefficients. To determine which coefficient changes are invisible to the data, apply the same expansion at a minimizer. The cross term vanishes, leaving

$$J(w^{\text{lin}} + v) - J(w^{\text{lin}}) = \frac{1}{2N} \|Xv\|_2^2. \quad (13)$$

A perturbation increases the objective precisely when it changes the training predictions. This is the distinction between uniqueness of a prediction vector and uniqueness of its coefficients.

Dependence among features. If the columns of X are independent, $Xv = 0$ implies $v = 0$, and (13) shows that the minimizing parameter vector is unique. Conversely, any nonzero v satisfying $Xv = 0$ can be added to a minimizer without changing the fit. The set of such vectors is the null space $\ker X$, and the complete set of solutions is $w^{\text{lin}} + \ker X$.

A unit conversion gives a useful instance of this dependence. If one feature records area in square feet and another records the same area in square meters, then $x_2 = cx_1$ for a fixed conversion factor c . Their contribution to a prediction is $(w_1 + cw_2)x_1$. The data can determine the combined coefficient, but there is no observation that tells us how to divide it between w_1 and w_2 . Measuring the same area twice has not given us two independent pieces of information. Replacing them by $w_1 - ct$ and $w_2 + t$ leaves the predictions unchanged. When the relation holds for future inputs as well, those inputs receive the same predictions under every such replacement. A dependence that happens to hold only in the training sample offers no such assurance outside it: at a new input x , adding a null-space vector v changes the prediction by $v^\top x$.

Full column rank also guarantees that $X^\top X$ is invertible. Indeed, $X^\top Xv = 0$ implies $0 = v^\top X^\top Xv = \|Xv\|_2^2$, so independence gives $v = 0$. Multiplying the normal equations by the inverse produces

$$w^{\text{lin}} = (X^\top X)^{-1} X^\top y, \quad \hat{y} = Py, \quad P = X(X^\top X)^{-1} X^\top. \quad (14)$$

The matrix P is symmetric and satisfies $P^2 = P$, as direct multiplication verifies. The second identity says that projecting a vector already in the column space has no further effect. Without full column rank, an orthonormal basis still gives $P = \sum_k u_k u_k^\top$, so the projection itself does not depend on the existence of the inverse formula.

When coefficients are not unique, a solver still has to choose some representative. A common convention is the minimum-norm solution, denoted X^+y , where X^+ is the Moore–Penrose pseudoinverse. Under this convention, we choose the shortest parameter vector among those making the best training predictions. That choice resolves an algebraic ambiguity; it cannot create evidence about a feature contribution that the observations never distinguished.

4 Computing the fit

We finally have a well-defined task for a computer. We chose what to record, restricted the hypothesis class, decided how errors should count, and worked out what a minimizing fit must satisfy. Finding such a fit is the job of an optimizer. Here, that procedure seeks coefficients with sufficiently small squared error at an acceptable computational cost. The inverse formula is one algebraic description of the answer, but it does not require us to evaluate an inverse.

In computation we solve a least-squares system rather than explicitly form $(X^\top X)^{-1}$. Factorizations such as QR or SVD supply stable numerical methods, including ways to handle dependent columns. Their implementation deserves separate attention; the question here is what information an iterative method would need to approach the same fit.

For a large dataset, another possibility is to improve an initial guess using a succession of inexpensive corrections. If the current model predicts too much for most examples, lowering its intercept is a reasonable correction. Adjusting a slope is subtler, because it changes some predictions more than others and may improve one part of the sample while worsening another. We need to measure the net effect of a small coefficient change on the error we chose to minimize.

A partial derivative measures exactly this local effect: the rate of change of J when one parameter varies and the others remain fixed. For observation i , the residual is $\epsilon^{(i)}(w) = y^{(i)} - w^\top x^{(i)}$, so

$\partial \epsilon^{(i)} / \partial w_j = -x_j^{(i)}$. Applying the chain rule to its squared contribution gives $\partial[(\epsilon^{(i)})^2/2] / \partial w_j = -\epsilon^{(i)} x_j^{(i)}$. Summing the contributions yields

$$\frac{\partial J(w)}{\partial w_j} = \frac{1}{N} \sum_{i=1}^N (w^\top x^{(i)} - y^{(i)}) x_j^{(i)}, \quad j = 0, \dots, d. \quad (15)$$

For the intercept, this is the negative mean residual. For a slope, each negative residual is multiplied by the corresponding feature value. Collecting these derivatives into a column vector gives the gradient $\nabla J(w)$, of length $d+1$. We can perform this same accounting with a matrix product: first negate the residual vector to obtain $Xw - y$, then multiply by each feature column and sum. Dividing by N gives the average contribution:

$$\nabla J(w) = \frac{1}{N} X^\top (Xw - y). \quad (16)$$

Setting the gradient to zero recovers the normal equations, whose sufficiency for optimality has already been proved geometrically.

Each entry of the gradient now has an interpretation: it tells us how sensitive the training objective is to a small change in one coefficient. We would like to turn those separate sensitivities into a coordinated correction. To choose a direction for changing all coefficients together, consider a small displacement Δw . Its first-order effect on the objective is $\nabla J(w)^\top \Delta w$. Among displacements of a given Euclidean length ρ , the most negative inner product is $-\rho \|\nabla J(w)\|_2$, attained in the direction opposite to the gradient. The Cauchy–Schwarz inequality $|a^\top b| \leq \|a\|_2 \|b\|_2$ gives the bound, and taking $\Delta w = -\rho \nabla J(w) / \|\nabla J(w)\|_2$ attains it when the gradient is nonzero. The Euclidean norm is part of this argument: it specifies which displacements count as equally large.

Following the negative gradient uses those local sensitivities to make a correction. The positive learning rate α , also called the *step size*, scales the negative gradient to set the parameter change at each iteration. Repeating this decision gives gradient descent:

$$w^{[t+1]} = w^{[t]} - \alpha \nabla J(w^{[t]}) = w^{[t]} - \frac{\alpha}{N} X^\top (Xw^{[t]} - y), \quad (17)$$

where $\alpha > 0$ is the learning rate and $[t]$ indexes an iteration. Every coordinate on the right is evaluated at the same old parameter vector. Replacing coordinates one at a time and using each newly computed value in subsequent derivatives would change the procedure. At a nonzero gradient, the chosen direction gives a decrease for a sufficiently short step. The remaining question is how short it must be.

Curvature and the step size. A derivative predicts the effect of a very small change. A numerical update takes a finite step, during which the slope of the objective can itself change. We need to account for that change before trusting a chosen learning rate. For least squares there is no approximation to make: expanding $\|Xw - y + Xv\|_2^2$ separates the first-order effect from the extra error contributed by the size of the step:

$$J(w + v) = J(w) + \nabla J(w)^\top v + \frac{1}{2} v^\top H v, \quad H = \frac{1}{N} X^\top X. \quad (18)$$

The matrix H is the Hessian, the matrix of second partial derivatives. Its quadratic form measures curvature along a displacement, and the Hessian is positive semidefinite because $v^\top H v = \|Xv\|_2^2 / N \geq 0$ for every v . The nonnegative curvature tells us something useful about the search: a local minimum

cannot conceal a better fit on the far side of a hill. Convexity expresses this property in terms of the graph: a convex function lies below the straight chord joining any two of its points. To check it here, compare the objective along the segment joining parameter vectors u and v with the corresponding average of their objectives:

$$J((1-s)u + sv) = (1-s)J(u) + sJ(v) - \frac{s(1-s)}{2N} \|X(u-v)\|_2^2, \quad 0 \leq s \leq 1.$$

The last term can only lower the value along the segment. A convex objective may still have many minimizing parameters when some directions are flat, as happens with dependent columns.

Convexity does not say that every attempted step improves the fit. To check the step itself, write $g = \nabla J(w)$ and substitute the actual displacement $-\alpha g$ into (18). The resulting change in the objective is

$$J(w - \alpha g) - J(w) = -\alpha \|g\|_2^2 + \frac{\alpha^2}{2} g^\top H g. \quad (19)$$

The negative term is proportional to the step size, while the positive curvature term is proportional to its square. Thus increasing the learning rate eventually sacrifices the decrease, even though the direction remains downhill at the starting point. In one dimension this is already visible for $J(w) = \lambda(w - w_*)^2/2$ with $\lambda > 0$. The update multiplies the parameter error $w - w_*$ by $1 - \alpha\lambda$. Its magnitude shrinks precisely when $0 < \alpha\lambda < 2$; for $1 < \alpha\lambda < 2$, the iterates alternate around the minimizer while approaching it.

With several coefficients, a step that is safe in one direction can be too large in another. We need the largest curvature among unit-length directions. For our symmetric Hessian this is its largest eigenvalue, which can be characterized by

$$\lambda_{\max} = \max_{\|v\|_2=1} v^\top H v.$$

For any nonzero g , apply this definition to $g/\|g\|_2$ to obtain $g^\top H g \leq \lambda_{\max} \|g\|_2^2$. Substituting that bound into (19) shows that

$$J(w - \alpha g) - J(w) \leq -\alpha \left(1 - \frac{\alpha \lambda_{\max}}{2}\right) \|g\|_2^2.$$

Thus $0 < \alpha < 2/\lambda_{\max}$ guarantees a decrease whenever $g \neq 0$. The condition is governed by the steepest curvature. If another direction is very shallow, the same cautious step may make little progress there. The method can take a long time even though the problem has a perfectly well-defined best fit. We will not need a full convergence theorem to see why the feature units can matter.

Feature scaling. The curvature disparity may reflect something as arbitrary as the units in which measurements were recorded. Changing a feature's units changes these curvatures: a coefficient multiplying a measurement in kilograms has a different numerical scale from one multiplying the same measurement in grams, even though both can express the same predictions. To express a feature in units of its typical observed variation, first subtract its training mean and then divide by its training standard deviation. These quantities and the resulting standardized feature are

$$\mu_j = \frac{1}{N} \sum_i x_j^{(i)}, \quad s_j = \left(\frac{1}{N} \sum_i (x_j^{(i)} - \mu_j)^2 \right)^{1/2}, \quad \tilde{x}_j = \frac{x_j - \mu_j}{s_j}.$$

For $s_j > 0$, the transformed feature has training mean zero and empirical variance one. We should also check whether this numerical convenience changes what the model can predict. Substituting the standardized feature into the affine hypothesis function and collecting its constant and input-dependent terms gives

$$\tilde{w}_0 + \sum_j \tilde{w}_j \tilde{x}_j = \left(\tilde{w}_0 - \sum_j \frac{\tilde{w}_j \mu_j}{s_j} \right) + \sum_j \frac{\tilde{w}_j}{s_j} x_j.$$

Thus the corresponding coefficients in the original units are

$$w_0 = \tilde{w}_0 - \sum_j \frac{\tilde{w}_j \mu_j}{s_j}, \quad w_j = \frac{\tilde{w}_j}{s_j}. \quad (20)$$

We can express the same hypothesis function in the original units by transforming its coefficients, including the intercept. The possible predictions and their best attainable fit have stayed the same. What can change is how readily the optimizer finds that fit. This is a useful surprise: changing how we describe a measurement can make a computation easier without adding information to the measurement.

Standardization often helps when the feature units produce very different scales. It does not remove correlations or make dependent columns independent. Min–max scaling, $(x_j - a_j)/(b_j - a_j)$ using the training minimum and maximum, serves a different purpose: it places the observed training range in $[0, 1]$. A future value outside that range remains outside it after scaling. Neither transformation makes unusual observations harmless.

For a constant training feature, both the range and the standard deviation are zero. Such a feature supplies no new direction when an intercept is present and can be removed instead of divided by zero. The intercept itself is left as a column of ones. For any chosen transformation, its fitted constants are part of the model: a new input must use the training μ_j and s_j , rather than statistics recomputed on the new data. Otherwise it is being expressed in coordinates different from those in which the coefficients were fitted.

Using a subset of the observations. A full gradient averages N observation-wise contributions. If a dense X has $d + 1$ columns, evaluating it costs order $N(d + 1)$ operations per iteration, whereas forming $X^\top X$ costs order $N(d + 1)^2$ before solving the resulting system. For very large N , inspecting every observation before making even one correction can be expensive. A smaller batch can estimate the same average, allowing us to trade the precision of a correction for a cheaper update. For a batch B of m sampled observations, use

$$g_B(w) = \frac{1}{m} \sum_{i \in B} (w^\top x^{(i)} - y^{(i)}) x^{(i)}, \quad w_{\text{new}} = w - \alpha g_B(w).$$

If B is sampled uniformly afresh, independently of the current iterate, this estimate averages to the full gradient: $\mathbb{E}[g_B(w) \mid w] = \nabla J(w)$. Here $\mathbb{E}[\cdot \mid w]$ means an average over possible batches with the current parameters held fixed. A particular sampled update can nevertheless increase the full objective. A batch of one gives stochastic gradient descent; using all observations recovers the full gradient. Smaller batches make updates cheaper by accepting a noisier direction, so the deterministic decrease argument above no longer applies to every step.

5 What a prediction estimates

Computing the coefficients accurately does not remove uncertainty in the target. Seedlings with the same recorded growing conditions can still reach different heights. Our hypothesis function nevertheless returns a single number. What should that number represent? Squared error gives us a precise answer once we describe the variation in target values at a fixed input. Let Y be a random target and $\mathbf{x}$ a random augmented input. For a fixed input value x , the conditional distribution of Y describes the possible target values and their probabilities given $\mathbf{x} = x$. It allows us to reason about the uncertainty that remains after the features are known.

We summarize these target values by their expectation, a probability-weighted average. For a discrete target it has the form $\mathbb{E}[Y] = \sum_y y \Pr(Y = y)$; for a continuous target with density p , the corresponding average is $\mathbb{E}[Y] = \int yp(y) dy$. Variance, $\text{Var}(Y) = \mathbb{E}[(Y - \mathbb{E}[Y])^2]$, measures average squared deviation from that mean. Their conditional versions use the distribution of target values at the given input x . Assuming a finite conditional second moment, let $m(x) = \mathbb{E}[Y \mid \mathbf{x} = x]$ be the mean target among cases with input x . To find which prediction minimizes the expected squared loss at this input, compare any proposed value a with $m(x)$. The squared error separates as follows:

$$\begin{aligned} \mathbb{E}[(Y - a)^2 \mid \mathbf{x} = x] &= \mathbb{E}[(Y - m(x))^2 \mid \mathbf{x} = x] \\ &\quad + 2(m(x) - a)\mathbb{E}[Y - m(x) \mid \mathbf{x} = x] + (m(x) - a)^2 \\ &= \text{Var}(Y \mid \mathbf{x} = x) + (m(x) - a)^2. \end{aligned} \tag{21}$$

The cross term vanishes because deviations from the conditional mean average to zero. The variance term remains no matter which number we predict. Choosing $a = m(x)$ eliminates the extra error contributed by our choice of that number. So a squared-error prediction has a concrete meaning: it aims at the mean target among cases with this input. Individual target values can differ from that average without making the prediction a poor estimate of what it was meant to represent. We have not needed a Gaussian assumption to reach this conclusion.

The first term describes the variation in target values remaining at input x , often called aleatoric uncertainty. Even knowing the conditional mean exactly would leave this variation in target values. How much variation remains depends on which measurements the input includes: a richer description can reveal distinctions we had previously averaged over. The second term measures the error in the particular number a we chose. Limited data can contribute to that error, but so can a hypothesis class that cannot represent the conditional mean. The decomposition helps separate the questions without pretending that every error has the same cause.

Linear regression approximates that conditional mean with an affine function of the supplied features. If $m(x)$ is not affine, the class cannot recover it at every input, regardless of how accurately the training objective is minimized. Even when it is affine, the observations provide only a finite sample from which to estimate its coefficients. Uncertainty about the relationship estimated from that finite sample is called epistemic uncertainty. That uncertainty differs from the target variation that remains even when the relationship is known. The choice of features, the choice of hypothesis class, and the amount of training data therefore limit prediction in different ways.

A probabilistic interpretation of squared error. There is another way to motivate our fitting criterion. Suppose each target fluctuates around the linear prediction according to a Gaussian noise model, independently across observations and with a shared variance $\sigma^2 > 0$:

$$Y^{(i)} = w^\top x^{(i)} + \varepsilon^{(i)}, \quad \varepsilon^{(i)} \mid X \stackrel{\text{ind}}{\sim} \mathcal{N}(0, \sigma^2). \tag{22}$$

Here $\varepsilon^{(i)}$ is the random *noise* in the assumed data-generating model. This noise is distinct from a residual, which we compute from an observed target and a chosen hypothesis function. Under the assumed model, the noise density at $y^{(i)} - w^\top x^{(i)}$ is proportional to $\exp[-(y^{(i)} - w^\top x^{(i)})^2 / (2\sigma^2)]$, with normalizing factor $1/\sqrt{2\pi\sigma^2}$. A density assigns probabilities to intervals by integration; its value at an exact real number is not itself a probability.

Holding the design X fixed, independence makes the joint density the product of those individual contributions. Evaluated at the observed targets, it is a function of the candidate parameters, called the likelihood:

$$L(w, \sigma^2; X, y) = (2\pi\sigma^2)^{-N/2} \exp\left(-\frac{1}{2\sigma^2} \text{RSS}(w)\right). \quad (23)$$

Maximum likelihood chooses parameters that give the observed data the largest joint density within the assumed model. Since the logarithm is increasing, we can equivalently maximize

$$\ell(w, \sigma^2) = -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \text{RSS}(w). \quad (24)$$

For fixed positive σ^2 , only the negative RSS term depends on w . Maximizing likelihood is therefore exactly the least-squares problem we have solved. Under this interpretation, the fitted hypothesis function gives the means of bell-shaped target distributions at different inputs.

The connection gives us something useful to question. If the observed spread increases with the predicted height, a common variance may be inappropriate. If the noise has known unequal variances, likelihood instead weights each squared residual by the inverse of the corresponding noise variance. Least squares still solves its stated optimization problem when the Gaussian assumptions fail; those assumptions determine which probabilistic conclusions we can draw from the answer. We chose both a hypothesis class and a way to judge its members, and each choice deserves to be examined against the observations.

6 Learning beyond the observed sample

The reason for fitting was to make an observation useful beyond the case in which it occurred. A small training error does not yet establish that we have done this. The selected features may miss a relationship, the hypothesis class may be too restrictive, or a limited sample may give a misleading estimate of its best coefficients. The sampled plants may differ systematically from those for which we intend to predict. Running an optimizer to convergence resolves the fitting problem for the supplied data; these questions concern the evidence supplied to that problem and the use we make of its solution.

To make the intended use explicit, imagine repeatedly encountering a new example from the population we care about and measuring the squared loss of the fitted hypothesis function. The average over that population is the *population risk*. It uses the same loss as the empirical risk, but averages over the intended population instead of the finite training dataset. For a fixed hypothesis function h_w , assuming finite expectations, averaging (21) over those new inputs gives

$$R(w) = \mathbb{E}[(Y - h_w(\mathbf{x}))^2] = \mathbb{E}[\text{Var}(Y \mid \mathbf{x})] + \mathbb{E}[(m(\mathbf{x}) - h_w(\mathbf{x}))^2]. \quad (25)$$

The first expectation is target variation left over after the recorded input is known. The second measures how closely the hypothesis function follows the conditional mean in the population where it will be used. For a fitted w , which is random through its training data, equation (25) applies to the

resulting function with that fit held fixed. We can estimate its performance using fresh observations sampled from the intended population. A random held-out subset of plants from one variety still evaluates prediction for that variety; splitting those records does not establish performance on another variety. Evaluating on the data that selected w instead measures how closely the fitting procedure matched its own sample.

The simplest empirical measures are RSS, MSE, and $\text{RMSE} = \sqrt{\text{MSE}}$. Taking that square root gives RMSE the same units as the target, so it is often easier to interpret as a scale of prediction error. To ask how much the recorded features helped, compare the total squared loss with that of a constant hypothesis function that always predicts the mean. Writing $\bar{y} = N^{-1} \sum_i y^{(i)}$ for the evaluation set's mean target, that baseline has total squared loss $\text{TSS} = \sum_i (y^{(i)} - \bar{y})^2$. Dividing the fitted function's RSS by TSS gives the fraction left over; subtracting it from one gives the relative reduction, denoted

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}, \quad \text{TSS} > 0. \quad (26)$$

The mean, residuals, and sums in this ratio must all refer to the same set of observations. An R^2 of 0.6 means that the total squared loss is 60 percent lower than that of always predicting the evaluation set's mean. This is a reported score, with larger values indicating improvement over that baseline; our training objective remains J . The ratio says nothing about the percentage of predictions that are individually correct, and it does not assign a causal meaning to the features.

For the training least-squares fit with an intercept, we can identify exactly which variation was captured. Subtracting the mean from the targets leaves their departures from the baseline. Each departure is the fitted departure plus a residual, and residual orthogonality prevents these two contributions from overlapping in squared length. The vector $\hat{y} - \bar{y}\mathbf{1}$ belongs to $\text{col}(X)$, so it is orthogonal to $y - \hat{y}$. Consequently,

$$\underbrace{\|y - \bar{y}\mathbf{1}\|_2^2}_{\text{TSS}} = \underbrace{\|\hat{y} - \bar{y}\mathbf{1}\|_2^2}_{\text{fitted variation}} + \underbrace{\|y - \hat{y}\|_2^2}_{\text{RSS}}. \quad (27)$$

On this sample, R^2 is the fraction of total squared variation represented by the fitted predictions. Both terms on the right are nonnegative, so $0 \leq R^2 \leq 1$. This conclusion uses an exact, unregularized fit with an intercept. On held-out data, or for a fit that lacks these properties, RSS may exceed TSS and R^2 may be negative. When all evaluation targets are identical, TSS is zero and the ratio is undefined.

Reading the residuals. A single error measure can hide a systematic failure. Suppose the input distribution is symmetric about zero and its conditional mean target is x_1^2 . On a symmetric sample with targets equal to those conditional means, the centered target has zero inner product with x_1 , because contributions at opposite inputs cancel. The best linear slope is then zero, although the target plainly depends on the input. The residuals trace a curve when plotted against x_1 . Orthogonality has removed the available linear component without removing the nonlinear relationship. A column containing x_1^2 would make that relationship representable.

A residual spread that grows with the fitted height suggests that the common-variance assumption deserves another look. An isolated large residual might reflect a recording error or a real but unusual target value. These are clues to investigate, not diagnoses delivered by the plot. The data collection matters as much as the algebra: repeated measurements of the same plant should not be split casually between training and evaluation, where they could make the evaluation look easier than predicting for new plants. All scaling constants must also be estimated from the training portion alone.

A line can be evaluated at twenty hours of daily light even if all our measurements were between six and ten. The arithmetic has no objection. The evidence is much less accommodating. A linear trend over a limited range need not continue indefinitely, and a held-out sample from that same narrow range cannot check the extrapolation.

Constructing the columns. The quadratic column above illustrates why a linear model can express a nonlinear function of a raw measurement. More generally, we can choose functions $\phi_1, \dots, \phi_d$ of a raw input z and predict $w_0 + \sum_j w_j \phi_j(z)$. The least-squares derivation applies once these functions have been evaluated to form the design matrix. Their choice determines which relationships are available to the fit. Polynomial powers allow curvature; interactions such as the product of two measurements allow the slope in one measurement to depend on the other.

Adding columns enlarges the set of possible training predictions, so its smallest RSS cannot increase. For nonconstant targets, training R^2 cannot decrease. This is why a smaller training loss alone cannot justify a more elaborate representation. We can always give the fit more freedom; the harder question is whether that freedom helps on observations it has not already seen. Model selection is the task of making that choice using evidence beyond the fit itself.

We began with many hypothesis functions that could agree on what we had seen and disagree on what came next. Restricting the hypothesis class tied those unseen predictions to a smaller set of choices. Least squares selected among them; geometry explained the selection; optimization made it computable. None of those steps alone established that the pattern would persist. That is what new observations let us investigate. The next batch of plants, like the next morning on the subway, gives the learned function a chance to be useful and gives us a chance to discover what it has missed.

Notation

Vectors are columns unless transposed; plain italic type is used for both vectors and scalars, with dimension determining the meaning. Parenthesized superscripts (i) index observations, square-bracketed superscripts $[t]$ index optimization steps, and unparenthesized superscripts such as x_1^2 denote powers. In the opening single-feature example, $x^{(i)}$ is a scalar and needs no feature subscript. From Section 2 onward, x and $x^{(i)}$ include the leading constant feature. A prime is not used for transpose.

Symbol	Meaning
$\mathcal{D}, \mathcal{H}$	Paired training dataset and hypothesis class (a set of functions).
N, d	Number of training observations and number of non-intercept features.
i, j, t	Observation, feature, and iteration indices.
$x_j^{(i)}$	Feature j of observation i ; $x_0^{(i)} = 1$ after augmentation.
$x^{(i)} \in \mathbb{R}^{d+1}$	Augmented input column $(1, x_1^{(i)}, \dots, x_d^{(i)})^\top$. Raw inputs have d coordinates.
$y^{(i)}, y \in \mathbb{R}^N$	Observed scalar target and stacked target vector.
$X \in \mathbb{R}^{N \times (d+1)}$	Augmented design matrix with row i equal to $(x^{(i)})^\top$.
$w \in \mathbb{R}^{d+1}$	Parameter vector: the intercept w_0 and slope coefficients.

Symbol	Meaning
$h, h_w, \hat{h}$	Hypothesis function, parameterized function, and fitted function.
$h_w(x) = w^\top x$	Prediction at augmented input x , also written $\hat{y}^{(i)}$ when $x = x^{(i)}$.
w^{lin}	A least-squares minimizing parameter vector.
$\hat{y} = Xw^{\text{lin}}$	Fitted training predictions; Xw denotes predictions at a general candidate w .
$\epsilon^{(i)}(w)$	Residual $y^{(i)} - h_w(x^{(i)})$ for a candidate fit.
$\hat{\epsilon}$	Fitted residual vector $y - Xw^{\text{lin}}$.
$\epsilon^{(i)}$	Random noise in a data-generating model, distinct from a fitted residual.
$\mathcal{L}(y, \hat{y})$	Loss on one observation; here $(y - \hat{y})^2$.
$\hat{R}_{\mathcal{D}}(h)$	Empirical risk: mean loss over the training dataset.
$\arg \min$	Set of arguments attaining the minimum of a function.
RSS, E_{in}	Residual sum of squares and mean training squared error, respectively.
$J(w)$	Optimization objective $\text{RSS}(w)/(2N) = E_{\text{in}}(w)/2$.
MSE, RMSE	Mean squared error and its square root on the specified evaluation set.
$\bar{y}$, TSS	Mean target and total squared deviation from that mean on the specified set.
$\alpha, w^{[t]}$	Learning rate (step size) and parameter vector at step t .
$\nabla J, H$	Gradient and Hessian; for least squares $H = X^\top X/N$.
μ_j, s_j	Training feature mean and standard deviation, using denominator N .
$\text{col}(X), \text{ker } X$	Column space and null space of X .
P, X^+	Orthogonal projection onto $\text{col}(X)$ and Moore–Penrose pseudoinverse.
$\ v\ _2, a^\top b$	Euclidean norm and dot product (Euclidean inner product).
$I, \mathbf{1}$	Identity matrix and all-ones vector, with dimensions given by context.
$\mathbb{E}, \text{Var}$	Expectation and variance.
x, Y	Random new augmented input and target; X remains the design matrix.
σ^2	Common noise variance in the Gaussian model.
$\mathcal{N}(0, \sigma^2)$	Gaussian distribution with zero mean and variance σ^2 .
L, ℓ	Likelihood and its natural logarithm; arguments state the parameters and data.
$R(w), R^2$	Population squared-error risk and coefficient of determination (distinct quantities).
$\lambda_{\max}$	Largest eigenvalue of the least-squares Hessian H .