CHEM 3PC3 - Mathematical Tools for Chemical Problems

Alexandre de Camargo

November 18, 2026

Inverting a matrix by elimination

- The inverse satisfies $AA^{-1} = I$. Write A^{-1} by its columns:

$$A \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix}$$

Inverting a matrix by elimination

- The inverse satisfies $AA^{-1} = I$. Write A^{-1} by its columns:

$$A \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix}$$

- So finding A^{-1} means solving **three** systems at once: $A\mathbf{x}_1 = \mathbf{e}_1$, $A\mathbf{x}_2 = \mathbf{e}_2$, $A\mathbf{x}_3 = \mathbf{e}_3$.

Inverting a matrix by elimination

- The inverse satisfies $AA^{-1} = I$. Write A^{-1} by its columns:

$$A \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix}$$

- So finding A^{-1} means solving **three** systems at once: $A\mathbf{x}_1 = \mathbf{e}_1$, $A\mathbf{x}_2 = \mathbf{e}_2$, $A\mathbf{x}_3 = \mathbf{e}_3$.
- They share the same matrix A, so we eliminate once and carry all three right hand sides along. That is the **Gauss–Jordan** method.

Set up the augmented matrix $[\,A \mid I\,]$

- Take $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 3 \\ 1 & 0 & 8 \end{bmatrix}$ and put the identity next to it:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 5 & 3 & 0 & 1 & 0 \\ 1 & 0 & 8 & 0 & 0 & 1 \end{array}\right]$$

Set up the augmented matrix $[\,A \mid I\,]$

- Take $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 3 \\ 1 & 0 & 8 \end{bmatrix}$ and put the identity next to it:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 5 & 3 & 0 & 1 & 0 \\ 1 & 0 & 8 & 0 & 0 & 1 \end{array}\right]$$

- The plan: row operations until the left block becomes I. Whatever appears on the right is A^{-1}.

Forward step 1: clear below the first pivot

- First pivot is 1. Multipliers $\ell_{21} = 2$ and $\ell_{31} = 1$.
- Do $R_2 - 2R_1$ and $R_3 - R_1$:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & -2 & 5 & -1 & 0 & 1 \end{array}\right]$$

Forward step 1: clear below the first pivot

- First pivot is 1. Multipliers $\ell_{21} = 2$ and $\ell_{31} = 1$.
- Do $R_2 - 2R_1$ and $R_3 - R_1$:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & -2 & 5 & -1 & 0 & 1 \end{array}\right]$$

- Notice the right block is no longer the identity — it is recording everything we did.

Forward step 2: clear below the second pivot

- Second pivot is 1 (row 2, column 2). Multiplier $\ell_{32} = -2$.
- Do $R_3 + 2R_2$:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & -1 & -5 & 2 & 1 \end{array}\right]$$

Forward step 2: clear below the second pivot

- Second pivot is 1 (row 2, column 2). Multiplier $\ell_{32} = -2$.
- Do $R_3 + 2R_2$:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & -1 & -5 & 2 & 1 \end{array}\right]$$

- The left block is now upper triangular. The pivots are 1, 1, -1 — none of them zero, so A **is** invertible.

Forward step 2: clear below the second pivot

- Second pivot is 1 (row 2, column 2). Multiplier $\ell_{32} = -2$.
- Do $R_3 + 2R_2$:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & -1 & -5 & 2 & 1 \end{array}\right]$$

- The left block is now upper triangular. The pivots are 1, 1, -1 — none of them zero, so A **is** invertible.
- If a pivot had come out zero with nothing to exchange, A would have no inverse and we would stop here.

Make the last pivot equal to 1

- Multiply row 3 by -1:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

Make the last pivot equal to 1

- Multiply row 3 by -1:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & -3 & -2 & 1 & 0 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

- Now we go **upward**, clearing the entries *above* each pivot. This is the Jordan half of the method.

Backward step 1: clear above the third pivot

- Do $R_2 + 3R_3$ and $R_1 - 3R_3$:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 0 & -14 & 6 & 3 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

Backward step 1: clear above the third pivot

- Do $R_2 + 3R_3$ and $R_1 - 3R_3$:

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 0 & -14 & 6 & 3 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

- Column 3 of the left block is done.

Backward step 2: clear above the second pivot

- Do $R_1 - 2R_2$:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -40 & 16 & 9 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

Backward step 2: clear above the second pivot

- Do $R_1 - 2R_2$:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -40 & 16 & 9 \\ 0 & 1 & 0 & 13 & -5 & -3 \\ 0 & 0 & 1 & 5 & -2 & -1 \end{array}\right]$$

- The left block is I. We are done.

The inverse

$$A^{-1} = \begin{bmatrix} -40 & 16 & 9 \\ 13 & -5 & -3 \\ 5 & -2 & -1 \end{bmatrix}$$

The inverse

$$A^{-1} = \begin{bmatrix} -40 & 16 & 9 \\ 13 & -5 & -3 \\ 5 & -2 & -1 \end{bmatrix}$$

- Check the first row of AA^{-1}:

$$\begin{aligned} (1,2,3)\cdot(-40,13,5) &= -40+26+15=1 \quad \checkmark \\ (1,2,3)\cdot(16,-5,-2) &= 16-10-6=0 \quad \checkmark \\ (1,2,3)\cdot(9,-3,-1) &= 9-6-3=0 \quad \checkmark \end{aligned}$$

The inverse

$$A^{-1} = \begin{bmatrix} -40 & 16 & 9 \\ 13 & -5 & -3 \\ 5 & -2 & -1 \end{bmatrix}$$

- Check the first row of AA^{-1}:

$$\begin{aligned} (1,2,3)\cdot(-40,13,5) &= -40+26+15=1 \quad \checkmark \\ (1,2,3)\cdot(16,-5,-2) &= 16-10-6=0 \quad \checkmark \\ (1,2,3)\cdot(9,-3,-1) &= 9-6-3=0 \quad \checkmark \end{aligned}$$

- The other two rows work the same way.

Why this works

- Every row operation is a multiplication on the left by some matrix. Call the product of all of them M. Then the process turns

$$[\,A \mid I\,] \quad \longrightarrow \quad [\,MA \mid M\,]$$

Why this works

- Every row operation is a multiplication on the left by some matrix. Call the product of all of them M. Then the process turns

$$[\,A \mid I\,] \quad \longrightarrow \quad [\,MA \mid M\,]$$

- We stopped exactly when $MA = I$. But $MA = I$ means $M = A^{-1}$.

Why this works

- Every row operation is a multiplication on the left by some matrix. Call the product of all of them M. Then the process turns

$$[\,A \mid I\,] \quad \longrightarrow \quad [\,MA \mid M\,]$$

- We stopped exactly when $MA = I$. But $MA = I$ means $M = A^{-1}$.
- So the right block *had* to come out as the inverse. Nothing else could have appeared there.

A warning about cost

- Gauss–Jordan on an $n \times n$ matrix costs about n^3 operations. Plain elimination to solve one system costs about $n^3/3$.

A warning about cost

- Gauss–Jordan on an $n \times n$ matrix costs about n^3 operations. Plain elimination to solve one system costs about $n^3/3$.
- So inverting and then multiplying, $\mathbf{x} = A^{-1}\mathbf{b}$, is roughly three times the work of solving $A\mathbf{x} = \mathbf{b}$ directly — and it loses accuracy.

A warning about cost

- Gauss–Jordan on an $n \times n$ matrix costs about n^3 operations. Plain elimination to solve one system costs about $n^3/3$.
- So inverting and then multiplying, $\mathbf{x} = A^{-1}\mathbf{b}$, is roughly three times the work of solving $A\mathbf{x} = \mathbf{b}$ directly — and it loses accuracy.
- Compute the inverse when you actually want the inverse. To solve a system, use elimination (`np.linalg.solve`), not `np.linalg.inv`.

Fitting a model to data

- So far every system had the same number of equations and unknowns.

Fitting a model to data

- So far every system had the same number of equations and unknowns.
- In the lab the opposite happens: we measure many points and fit a model with only a few parameters. X has **more rows than columns**.

Fitting a model to data

- So far every system had the same number of equations and unknowns.
- In the lab the opposite happens: we measure many points and fit a model with only a few parameters. X has **more rows than columns**.
- Unless the measurements are perfect, $\hat{\mathbf{y}}$ is not a combination of the columns of X. Elimination hits an impossible equation and stops.

Fitting a model to data

- So far every system had the same number of equations and unknowns.
- In the lab the opposite happens: we measure many points and fit a model with only a few parameters. X has **more rows than columns**.
- Unless the measurements are perfect, $\hat{\mathbf{y}}$ is not a combination of the columns of X. Elimination hits an impossible equation and stops.
- But we cannot throw away data just because it has noise. We look for the $\mathbf{w}$ that makes the error as small as possible.

The example: fit a straight line

- Three measurements $(x_i, \hat{y}_i)$: $(0, 2)$, $(1, 1)$, $(2, 6)$.

The example: fit a straight line

- Three measurements $(x_i, \hat{y}_i)$: $(0, 2)$, $(1, 1)$, $(2, 6)$.
- The model is $f(x, \mathbf{w}) = w_0 + w_1 x$.

The example: fit a straight line

- Three measurements $(x_i, \hat{y}_i)$: $(0, 2)$, $(1, 1)$, $(2, 6)$.
- The model is $f(x, \mathbf{w}) = w_0 + w_1 x$.
- If the line passed through all three points, w_0 and w_1 would satisfy

$$\begin{cases} w_0 + w_1 \cdot 0 = 2 \\ w_0 + w_1 \cdot 1 = 1 \\ w_0 + w_1 \cdot 2 = 6 \end{cases}$$

The example: fit a straight line

- Three measurements $(x_i, \hat{y}_i)$: $(0, 2)$, $(1, 1)$, $(2, 6)$.
- The model is $f(x, \mathbf{w}) = w_0 + w_1 x$.
- If the line passed through all three points, w_0 and w_1 would satisfy

$$\begin{cases} w_0 + w_1 \cdot 0 = 2 \\ w_0 + w_1 \cdot 1 = 1 \\ w_0 + w_1 \cdot 2 = 6 \end{cases}$$

- Three equations, two unknowns. No line goes through all three points.

In matrix form

- Each measurement gives a feature vector $\mathbf{x}_i = \Phi(x_i) = (1, x_i)$, and these become the **rows** of X:

$$X = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \qquad \hat{\mathbf{y}} = \begin{bmatrix} 2 \\ 1 \\ 6 \end{bmatrix}$$

In matrix form

- Each measurement gives a feature vector $\mathbf{x}_i = \Phi(x_i) = (1, x_i)$, and these become the **rows** of X:

$$X = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \qquad \hat{\mathbf{y}} = \begin{bmatrix} 2 \\ 1 \\ 6 \end{bmatrix}$$

- The three equations are $X\mathbf{w} = \hat{\mathbf{y}}$, the same shape as $A\mathbf{x} = \mathbf{b}$ from the elimination slides.

In matrix form

- Each measurement gives a feature vector $\mathbf{x}_i = \Phi(x_i) = (1, x_i)$, and these become the **rows** of X:

$$X = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \qquad \hat{\mathbf{y}} = \begin{bmatrix} 2 \\ 1 \\ 6 \end{bmatrix}$$

- The three equations are $X\mathbf{w} = \hat{\mathbf{y}}$, the same shape as $A\mathbf{x} = \mathbf{b}$ from the elimination slides.
- But X is 3×2 and this system is **not solvable**. The columns $(1, 1, 1)$ and $(0, 1, 2)$ span a plane, and $\hat{\mathbf{y}} = (2, 1, 6)$ is not in it.

What do we minimize?

- The prediction at point i is $y_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i$, and the residual is $\varepsilon_i = \hat{y}_i - y_i$.

What do we minimize?

- The prediction at point i is $y_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i$, and the residual is $\varepsilon_i = \hat{y}_i - y_i$.
- We cannot make every ε_i zero. Instead we minimize the **mean squared error**:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left(\hat{y}_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)^2 = \frac{1}{2N}\left\|\hat{\mathbf{y}} - X\mathbf{w}\right\|^2$$

What do we minimize?

- The prediction at point i is $y_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i$, and the residual is $\varepsilon_i = \hat{y}_i - y_i$.
- We cannot make every ε_i zero. Instead we minimize the **mean squared error**:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left(\hat{y}_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)^2 = \frac{1}{2N}\left\|\hat{\mathbf{y}} - X\mathbf{w}\right\|^2$$

- The ε_i are the *vertical* distances from each point to the line.

What do we minimize?

- The prediction at point i is $y_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i$, and the residual is $\varepsilon_i = \hat{y}_i - y_i$.
- We cannot make every ε_i zero. Instead we minimize the **mean squared error**:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left(\hat{y}_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)^2 = \frac{1}{2N}\left\|\hat{\mathbf{y}} - X\mathbf{w}\right\|^2$$

- The ε_i are the *vertical* distances from each point to the line.
- Squaring weights the big misses much more heavily than the small ones.

The Jacobian, one component at a time

- The loss is $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \right)^2$.

The Jacobian, one component at a time

- The loss is $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \right)^2$.
- Differentiate with respect to one parameter w_j. The chain rule gives

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \right) \frac{\partial}{\partial w_j} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \right)$$

The Jacobian, one component at a time

- The loss is $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \right)^2$.
- Differentiate with respect to one parameter w_j. The chain rule gives

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \right) \frac{\partial}{\partial w_j} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \right)$$

- The $\frac{1}{2}$ cancels the 2 from the square.

The Jacobian, one component at a time

- The loss is $\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i\right)^2$.
- Differentiate with respect to one parameter w_j. The chain rule gives

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i\right) \frac{\partial}{\partial w_j} \left(\hat{y}_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i\right)$$

- The $\frac{1}{2}$ cancels the 2 from the square.
- Now we only need $\frac{\partial}{\partial w_j} \left(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i\right)$.

The inner derivative

- Write the dot product as a sum:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i = \sum_{k=0}^{d} w_k\, x_{ik} = w_0 x_{i0} + w_1 x_{i1} + \cdots + w_d x_{id}$$

The inner derivative

- Write the dot product as a sum:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i = \sum_{k=0}^{d} w_k\, x_{ik} = w_0 x_{i0} + w_1 x_{i1} + \cdots + w_d x_{id}$$

- Only the term $k = j$ contains w_j. Every other term is a constant:

$$\frac{\partial}{\partial w_j}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right) = x_{ij}$$

The inner derivative

- Write the dot product as a sum:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i = \sum_{k=0}^{d} w_k\, x_{ik} = w_0 x_{i0} + w_1 x_{i1} + \cdots + w_d x_{id}$$

- Only the term $k = j$ contains w_j. Every other term is a constant:

$$\frac{\partial}{\partial w_j}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right) = x_{ij}$$

- So the derivative of the model with respect to a parameter is just the feature that parameter multiplies. This is the whole reason linear models are easy.

Putting it together

- Substituting back, and keeping the minus sign:

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right) x_{ij} = -\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i\, x_{ij}$$

Putting it together

- Substituting back, and keeping the minus sign:

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right) x_{ij} = -\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i\, x_{ij}$$

- The Jacobian collects all of these into one vector:

$$\nabla_{\mathbf{w}}\mathcal{L} = \begin{bmatrix} \partial\mathcal{L}/\partial w_0 \\ \vdots \\ \partial\mathcal{L}/\partial w_d \end{bmatrix} = -\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i\,\mathbf{x}_i$$

From the sum to the matrix

- The feature vectors $\mathbf{x}_i$ are the **rows** of X, so they are the **columns** of X^{T}:

$$X^{\mathrm{T}}\boldsymbol{\varepsilon} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix} = \sum_{i=1}^{N} \varepsilon_i \, \mathbf{x}_i$$

From the sum to the matrix

- The feature vectors $\mathbf{x}_i$ are the **rows** of X, so they are the **columns** of X^{T}:

$$X^{\mathrm{T}}\boldsymbol{\varepsilon} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix} = \sum_{i=1}^{N} \varepsilon_i \, \mathbf{x}_i$$

- A matrix times a vector is a combination of the columns — exactly the sum we just wrote.

From the sum to the matrix

- The feature vectors $\mathbf{x}_i$ are the **rows** of X, so they are the **columns** of X^{T}:

$$X^{\mathrm{T}}\boldsymbol{\varepsilon} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix} = \sum_{i=1}^{N} \varepsilon_i \, \mathbf{x}_i$$

- A matrix times a vector is a combination of the columns — exactly the sum we just wrote.
- Therefore

$$\nabla_{\mathbf{w}} \mathcal{L} = -\frac{1}{N} X^{\mathrm{T}} \boldsymbol{\varepsilon} = -\frac{1}{N} X^{\mathrm{T}} (\hat{\mathbf{y}} - X\mathbf{w})$$

The shortcut, all in matrices

- Expand the loss directly:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{2N}\left(\hat{\mathbf{y}} - X\mathbf{w}\right)^{\mathrm{T}}\left(\hat{\mathbf{y}} - X\mathbf{w}\right) \\ &= \frac{1}{2N}\left(\hat{\mathbf{y}}^{\mathrm{T}}\hat{\mathbf{y}} - 2\,\mathbf{w}^{\mathrm{T}}X^{\mathrm{T}}\hat{\mathbf{y}} + \mathbf{w}^{\mathrm{T}}X^{\mathrm{T}}X\mathbf{w}\right)\end{aligned}$$

The shortcut, all in matrices

- Expand the loss directly:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{2N}\left(\hat{\mathbf{y}} - X\mathbf{w}\right)^{\mathrm{T}}\left(\hat{\mathbf{y}} - X\mathbf{w}\right) \\ &= \frac{1}{2N}\left(\hat{\mathbf{y}}^{\mathrm{T}}\hat{\mathbf{y}} - 2\,\mathbf{w}^{\mathrm{T}}X^{\mathrm{T}}\hat{\mathbf{y}} + \mathbf{w}^{\mathrm{T}}X^{\mathrm{T}}X\mathbf{w}\right)\end{aligned}$$

- Two rules of vector calculus are enough:

$$\nabla_{\mathbf{w}}\left(\mathbf{w}^{\mathrm{T}}\mathbf{a}\right) = \mathbf{a} \qquad \nabla_{\mathbf{w}}\left(\mathbf{w}^{\mathrm{T}}S\mathbf{w}\right) = 2S\mathbf{w} \quad (S \text{ symmetric})$$

The shortcut, all in matrices

- Expand the loss directly:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{2N}\left(\hat{\mathbf{y}} - X\mathbf{w}\right)^{\mathrm{T}}\left(\hat{\mathbf{y}} - X\mathbf{w}\right) \\ &= \frac{1}{2N}\left(\hat{\mathbf{y}}^{\mathrm{T}}\hat{\mathbf{y}} - 2\,\mathbf{w}^{\mathrm{T}}X^{\mathrm{T}}\hat{\mathbf{y}} + \mathbf{w}^{\mathrm{T}}X^{\mathrm{T}}X\mathbf{w}\right)\end{aligned}$$

- Two rules of vector calculus are enough:

$$\nabla_{\mathbf{w}}\left(\mathbf{w}^{\mathrm{T}}\mathbf{a}\right) = \mathbf{a} \qquad \nabla_{\mathbf{w}}\left(\mathbf{w}^{\mathrm{T}}S\mathbf{w}\right) = 2S\mathbf{w} \quad (S \text{ symmetric})$$

- Since $X^{\mathrm{T}}X$ is symmetric,

$$\nabla_{\mathbf{w}}\mathcal{L} = \frac{1}{2N}\left(-2X^{\mathrm{T}}\hat{\mathbf{y}} + 2X^{\mathrm{T}}X\mathbf{w}\right) = \frac{1}{N}\left(X^{\mathrm{T}}X\mathbf{w} - X^{\mathrm{T}}\hat{\mathbf{y}}\right)$$

The minimum: set the gradient to zero

- $\mathcal{L}$ is a function of the parameters, so at the minimum every partial derivative vanishes. That vector of partials is the Jacobian $\nabla_{\mathbf{w}}\mathcal{L}$.

The minimum: set the gradient to zero

- $\mathcal{L}$ is a function of the parameters, so at the minimum every partial derivative vanishes. That vector of partials is the Jacobian $\nabla_{\mathbf{w}}\mathcal{L}$.

$$\nabla_{\mathbf{w}}\mathcal{L} \;=\; -\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)\mathbf{x}_i \;=\; -\frac{1}{N}\,X^{\mathrm{T}}(\hat{\mathbf{y}} - X\mathbf{w})$$

The minimum: set the gradient to zero

- $\mathcal{L}$ is a function of the parameters, so at the minimum every partial derivative vanishes. That vector of partials is the Jacobian $\nabla_{\mathbf{w}}\mathcal{L}$.

$$\nabla_{\mathbf{w}}\mathcal{L} \;=\; -\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i\right)\mathbf{x}_i \;=\; -\frac{1}{N}\,X^{\mathrm{T}}(\hat{\mathbf{y}} - X\mathbf{w})$$

- The $\frac{1}{2}$ in $\mathcal{L}$ existed only to cancel the 2 from the chain rule.

The normal equation

- Set $\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{0}$. The factor $-\frac{1}{N}$ drops out:

$$X^{\mathrm{T}}(\hat{\mathbf{y}} - X\mathbf{w}) = \mathbf{0}$$

The normal equation

- Set $\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{0}$. The factor $-\frac{1}{N}$ drops out:

$$X^{\mathrm{T}}(\hat{\mathbf{y}} - X\mathbf{w}) = \mathbf{0}$$

$$X^{\mathrm{T}} X\, \mathbf{w}^* = X^{\mathrm{T}} \hat{\mathbf{y}}$$

The normal equation

- Set $\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{0}$. The factor $-\frac{1}{N}$ drops out:

$$X^{\mathrm{T}}(\hat{\mathbf{y}} - X\mathbf{w}) = \mathbf{0}$$

$$X^{\mathrm{T}} X\, \mathbf{w}^* = X^{\mathrm{T}} \hat{\mathbf{y}}$$

- X is 3×2, so $X^{\mathrm{T}} X$ is 2×2 — a small square system we already know how to solve.

The normal equation

- Set $\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{0}$. The factor $-\frac{1}{N}$ drops out:

$$X^{\mathrm{T}}(\hat{\mathbf{y}} - X\mathbf{w}) = \mathbf{0}$$

$$X^{\mathrm{T}} X\, \mathbf{w}^* = X^{\mathrm{T}} \hat{\mathbf{y}}$$

- X is 3×2, so $X^{\mathrm{T}}X$ is 2×2 — a small square system we already know how to solve.
- When $X^{\mathrm{T}}X$ is invertible, $\mathbf{w}^* = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\hat{\mathbf{y}}$.

Step 1: build $X^{\mathrm{T}}X$ and $X^{\mathrm{T}}\hat{\mathbf{y}}$

$$X^{\mathrm{T}}X = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}$$

Step 1: build $X^{\mathrm{T}}X$ and $X^{\mathrm{T}}\hat{\mathbf{y}}$

$$X^{\mathrm{T}}X = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}$$

$$X^{\mathrm{T}}\hat{\mathbf{y}} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 6 \end{bmatrix} = \begin{bmatrix} 9 \\ 13 \end{bmatrix}$$

Step 1: build $X^{\mathrm{T}}X$ and $X^{\mathrm{T}}\hat{\mathbf{y}}$

$$X^{\mathrm{T}}X = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}$$

$$X^{\mathrm{T}}\hat{\mathbf{y}} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 6 \end{bmatrix} = \begin{bmatrix} 9 \\ 13 \end{bmatrix}$$

- Note that $X^{\mathrm{T}}X$ is symmetric — it always is.

Step 2: solve the 2×2 system

$$\begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 9 \\ 13 \end{bmatrix} \qquad \begin{cases} 3w_0 + 3w_1 = 9 \\ 3w_0 + 5w_1 = 13 \end{cases}$$

Step 2: solve the 2×2 system

$$\begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 9 \\ 13 \end{bmatrix} \qquad \begin{cases} 3w_0 + 3w_1 = 9 \\ 3w_0 + 5w_1 = 13 \end{cases}$$

- Subtract the first equation from the second:

$$2w_1 = 4 \qquad \Longrightarrow \qquad w_1 = 2$$

Step 2: solve the 2×2 system

$$\begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 9 \\ 13 \end{bmatrix} \qquad \begin{cases} 3w_0 + 3w_1 = 9 \\ 3w_0 + 5w_1 = 13 \end{cases}$$

- Subtract the first equation from the second:

$$2w_1 = 4 \qquad \Longrightarrow \qquad w_1 = 2$$

- Back substitute into $3w_0 + 3w_1 = 9$:

$$3w_0 + 6 = 9 \qquad \Longrightarrow \qquad w_0 = 1$$

Step 2: solve the 2×2 system

$$\begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 9 \\ 13 \end{bmatrix} \qquad \begin{cases} 3w_0 + 3w_1 = 9 \\ 3w_0 + 5w_1 = 13 \end{cases}$$

- Subtract the first equation from the second:

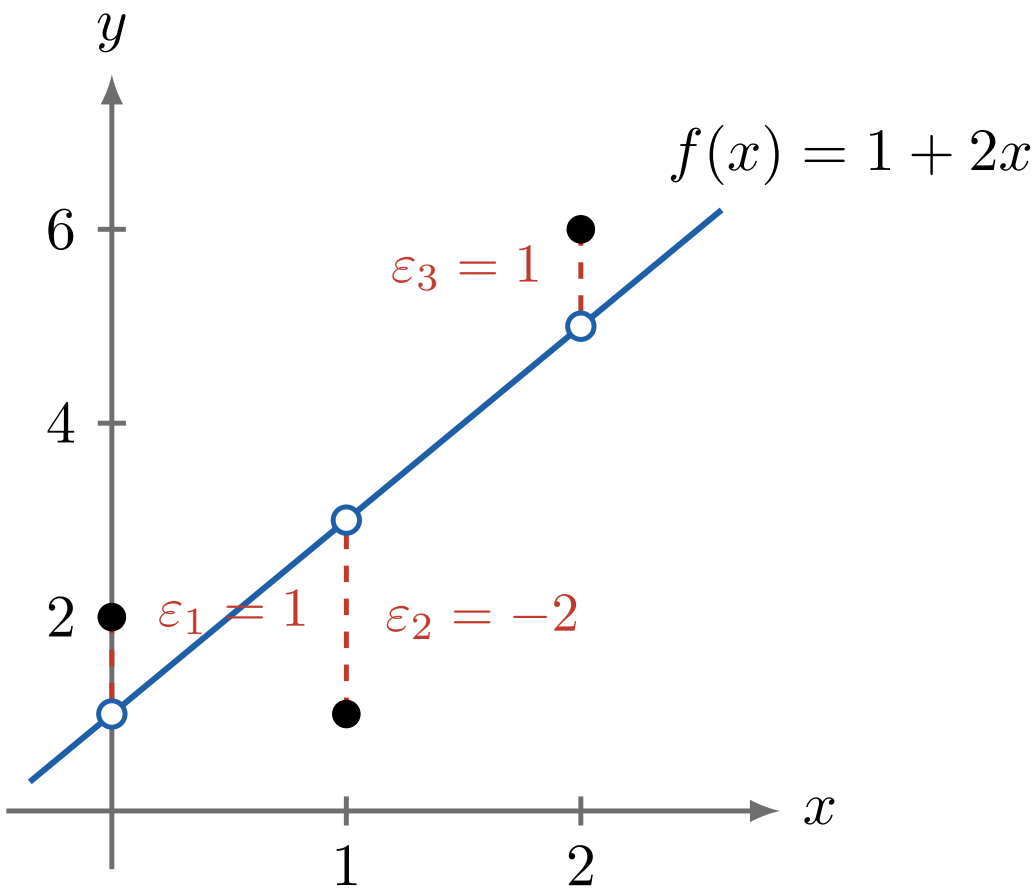
$$2w_1 = 4 \qquad \Longrightarrow \qquad w_1 = 2$$

- Back substitute into $3w_0 + 3w_1 = 9$:

$$3w_0 + 6 = 9 \qquad \Longrightarrow \qquad w_0 = 1$$

- So $\mathbf{w}^* = (1, 2)$ and the best model is $f(x) = 1 + 2x$.

The closest line



Predictions and residuals

- The predictions are $\mathbf{y} = X\mathbf{w}^*$:

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \qquad \boldsymbol{\varepsilon} = \hat{\mathbf{y}} - \mathbf{y} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Predictions and residuals

- The predictions are $\mathbf{y} = X\mathbf{w}^*$:

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \qquad \boldsymbol{\varepsilon} = \hat{\mathbf{y}} - \mathbf{y} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

- The smallest possible loss is

$$\mathcal{L}(\mathbf{w}^*) = \frac{1}{2N}\left(1^2 + (-2)^2 + 1^2\right) = \frac{6}{6} = 1$$

Predictions and residuals

- The predictions are $\mathbf{y} = X\mathbf{w}^*$:

$$\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \qquad \boldsymbol{\varepsilon} = \hat{\mathbf{y}} - \mathbf{y} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

- The smallest possible loss is

$$\mathcal{L}(\mathbf{w}^*) = \frac{1}{2N}\left(1^2 + (-2)^2 + 1^2\right) = \frac{6}{6} = 1$$

- Every measurement splits as $\hat{\mathbf{y}} = \mathbf{y} + \boldsymbol{\varepsilon}$: the part the model can reach, plus the part it cannot.

What the gradient really says

- At the minimum, $X^{\mathrm{T}}\boldsymbol{\varepsilon} = \mathbf{0}$.

What the gradient really says

- At the minimum, $X^{\mathrm{T}}\boldsymbol{\varepsilon} = \mathbf{0}$.
- **The residual is orthogonal to every column of X.** Check it with $\boldsymbol{\varepsilon} = (1, -2, 1)$:

$$\begin{aligned} (1,1,1)\cdot(1,-2,1) &= 1-2+1=0 \quad \checkmark \\ (0,1,2)\cdot(1,-2,1) &= 0-2+2=0 \quad \checkmark \end{aligned}$$

What the gradient really says

- At the minimum, $X^{\mathrm{T}}\boldsymbol{\varepsilon} = \mathbf{0}$.
- **The residual is orthogonal to every column of X.** Check it with $\boldsymbol{\varepsilon} = (1, -2, 1)$:

$$\begin{aligned} (1,1,1)\cdot(1,-2,1) &= 1-2+1=0 \quad \checkmark \\ (0,1,2)\cdot(1,-2,1) &= 0-2+2=0 \quad \checkmark \end{aligned}$$

- Geometrically, $\mathbf{y}$ is the point of the column plane closest to $\hat{\mathbf{y}}$, and $\boldsymbol{\varepsilon}$ is what is left over, sticking out perpendicularly.

The same answer from partial derivatives

- Write the loss out in full for our three points:

$$\mathcal{L} = \frac{1}{6}\left[(w_0 - 2)^2 + (w_0 + w_1 - 1)^2 + (w_0 + 2w_1 - 6)^2\right]$$

The same answer from partial derivatives

- Write the loss out in full for our three points:

$$\mathcal{L} = \frac{1}{6}\left[(w_0 - 2)^2 + (w_0 + w_1 - 1)^2 + (w_0 + 2w_1 - 6)^2\right]$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{3}\left[(w_0 - 2) + (w_0 + w_1 - 1) + (w_0 + 2w_1 - 6)\right] = 0$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{3}\left[(w_0 + w_1 - 1)(1) + (w_0 + 2w_1 - 6)(2)\right] = 0$$

The same answer from partial derivatives

- Write the loss out in full for our three points:

$$\mathcal{L} = \frac{1}{6}\left[(w_0 - 2)^2 + (w_0 + w_1 - 1)^2 + (w_0 + 2w_1 - 6)^2\right]$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{1}{3}\left[(w_0 - 2) + (w_0 + w_1 - 1) + (w_0 + 2w_1 - 6)\right] = 0$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{1}{3}\left[(w_0 + w_1 - 1)(1) + (w_0 + 2w_1 - 6)(2)\right] = 0$$

- Collecting terms gives $3w_0 + 3w_1 = 9$ and $3w_0 + 5w_1 = 13$ — the normal equations again.

The general formula for a straight line

- For N points the columns of X are $(1, \ldots, 1)$ and $(x_1, \ldots, x_N)$. Then

$$X^{\mathrm{T}}X = \begin{bmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \qquad X^{\mathrm{T}}\hat{\mathbf{y}} = \begin{bmatrix} \sum \hat{y}_i \\ \sum x_i \hat{y}_i \end{bmatrix}$$

The general formula for a straight line

- For N points the columns of X are $(1, \ldots, 1)$ and $(x_1, \ldots, x_N)$. Then

$$X^{\mathrm{T}}X = \begin{bmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \qquad X^{\mathrm{T}}\hat{\mathbf{y}} = \begin{bmatrix} \sum \hat{y}_i \\ \sum x_i \hat{y}_i \end{bmatrix}$$

- So the whole fit reduces to four sums:

$$\begin{bmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \sum \hat{y}_i \\ \sum x_i \hat{y}_i \end{bmatrix}$$

The general formula for a straight line

- For N points the columns of X are $(1, \ldots, 1)$ and $(x_1, \ldots, x_N)$. Then

$$X^{\mathrm{T}}X = \begin{bmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \qquad X^{\mathrm{T}}\hat{\mathbf{y}} = \begin{bmatrix} \sum \hat{y}_i \\ \sum x_i \hat{y}_i \end{bmatrix}$$

- So the whole fit reduces to four sums:

$$\begin{bmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \sum \hat{y}_i \\ \sum x_i \hat{y}_i \end{bmatrix}$$

- In our example: $N = 3$, $\sum x_i = 3$, $\sum x_i^2 = 5$, $\sum \hat{y}_i = 9$, $\sum x_i \hat{y}_i = 13$.

Two ways to reach $\mathbf{w}^*$

- **Closed form.** Solve $X^{\mathrm{T}} X \mathbf{w}^* = X^{\mathrm{T}} \hat{\mathbf{y}}$ once. This is what we just did.

Two ways to reach $\mathbf{w}^*$

- **Closed form.** Solve $X^{\mathrm{T}} X \mathbf{w}^* = X^{\mathrm{T}} \hat{\mathbf{y}}$ once. This is what we just did.
- **Gradient descent.** Start anywhere and step downhill:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} + \frac{\eta}{N} X^{\mathrm{T}} \boldsymbol{\varepsilon}$$

Two ways to reach $\mathbf{w}^*$

- **Closed form.** Solve $X^{\mathrm{T}} X \mathbf{w}^* = X^{\mathrm{T}} \hat{\mathbf{y}}$ once. This is what we just did.
- **Gradient descent.** Start anywhere and step downhill:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} + \frac{\eta}{N} X^{\mathrm{T}} \boldsymbol{\varepsilon}$$

- The update keeps moving until $X^{\mathrm{T}} \boldsymbol{\varepsilon} = \mathbf{0}$ — the same condition, reached one step at a time.

Two ways to reach $\mathbf{w}^*$

- **Closed form.** Solve $X^{\mathrm{T}} X \mathbf{w}^* = X^{\mathrm{T}} \hat{\mathbf{y}}$ once. This is what we just did.
- **Gradient descent.** Start anywhere and step downhill:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} + \frac{\eta}{N} X^{\mathrm{T}} \boldsymbol{\varepsilon}$$

- The update keeps moving until $X^{\mathrm{T}} \boldsymbol{\varepsilon} = \mathbf{0}$ — the same condition, reached one step at a time.
- For a linear model the closed form wins. But when the model is not linear in $\mathbf{w}$ — a neural network, for instance — there is no closed form and gradient descent is all we have.

Why polynomials are still linear models

- Fitting $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$ looks nonlinear, but it is linear **in the parameters**. Only the feature map Φ changes:

$$X_{\text{line}} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \qquad X_{\text{parabola}} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

Why polynomials are still linear models

- Fitting $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$ looks nonlinear, but it is linear **in the parameters**. Only the feature map Φ changes:

$$X_{\text{line}} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \qquad X_{\text{parabola}} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

- Same normal equation, same solve, one more column.

Why polynomials are still linear models

- Fitting $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$ looks nonlinear, but it is linear **in the parameters**. Only the feature map Φ changes:

$$X_{\text{line}} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \qquad X_{\text{parabola}} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

- Same normal equation, same solve, one more column.
- This is why the whole family is called **linear models**: every parameter w_i multiplies its feature exactly once.

In Python

```
import numpy as np

x     = np.array([0., 1., 2.])
y_hat = np.array([2., 1., 6.])

X = np.column_stack([np.ones_like(x), x])

w, res, rank, sv = np.linalg.lstsq(X, y_hat, rcond=None)
```

In Python

```
import numpy as np

x     = np.array([0., 1., 2.])
y_hat = np.array([2., 1., 6.])

X = np.column_stack([np.ones_like(x), x])

w, res, rank, sv = np.linalg.lstsq(X, y_hat, rcond=None)
```

- Output: `w = [1.  2.]`, the same w_0 and w_1.

In Python

```
import numpy as np

x     = np.array([0., 1., 2.])
y_hat = np.array([2., 1., 6.])

X = np.column_stack([np.ones_like(x), x])

w, res, rank, sv = np.linalg.lstsq(X, y_hat, rcond=None)
```

- Output: `w = [1.  2.]`, the same w_0 and w_1.
- `lstsq` never forms $X^{\mathrm{T}}X$ — it uses a more stable factorization. Use it instead of building the normal equations by hand.