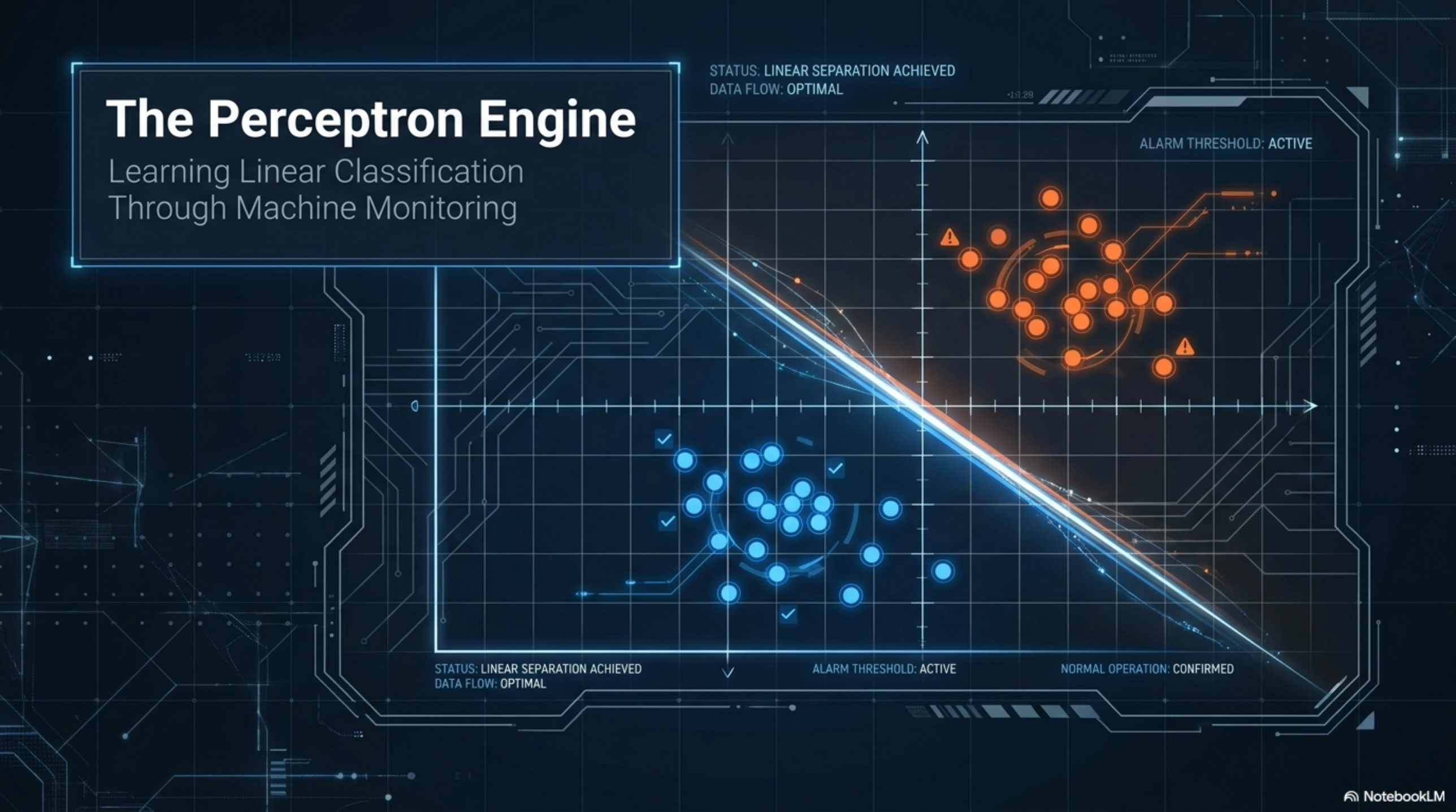
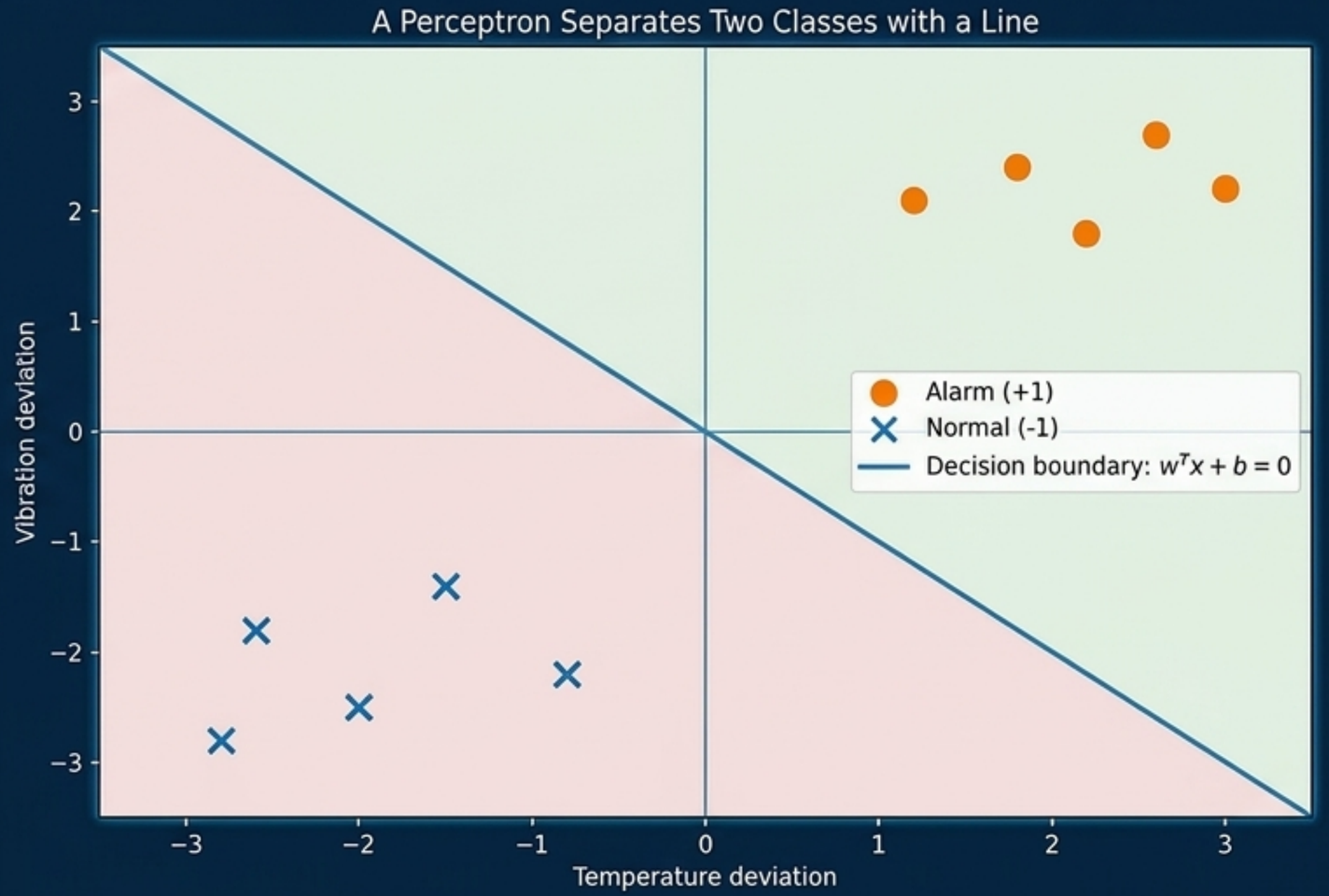
The Perceptron Engine

Learning Linear Classification Through Machine Monitoring



$$x = [2, 1]^T$$

Goal: Learn a rule from known examples to classify new machine states.



$$s = w_1 x_1 + w_2 x_2 + b$$

The Raw Telemetry Signal (Score)

The Sensor Readings (Features)

Direction & Strength (Weights)

How much one feature affects the final score. e.g., 1.5x_1 means temp has a stronger effect than 0.5x_2 vibration

The Boundary Shifter (Bias)

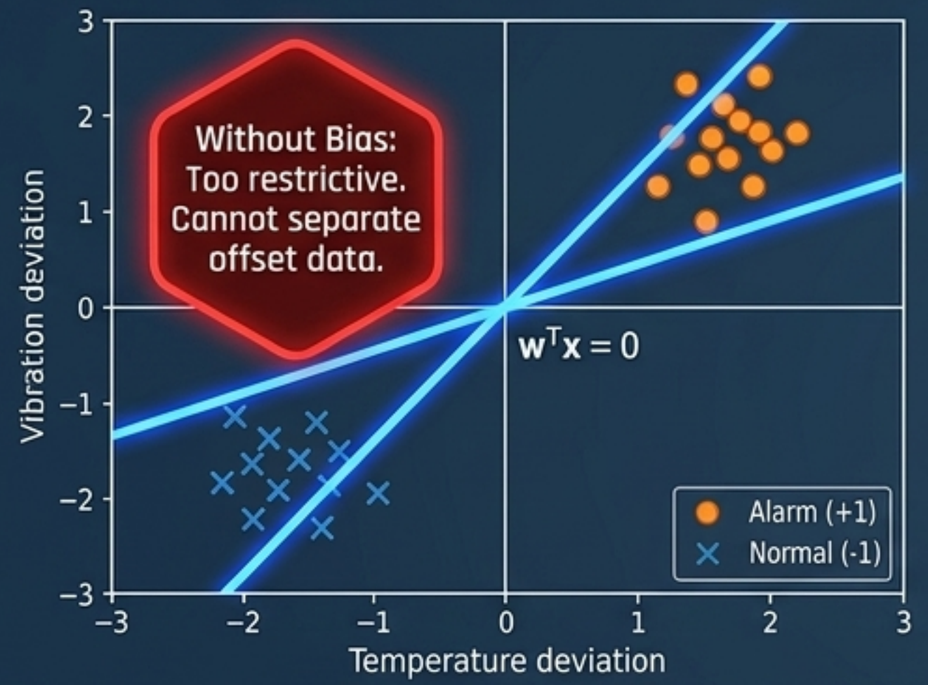
The Prediction Filter

If s > 0, output is +1

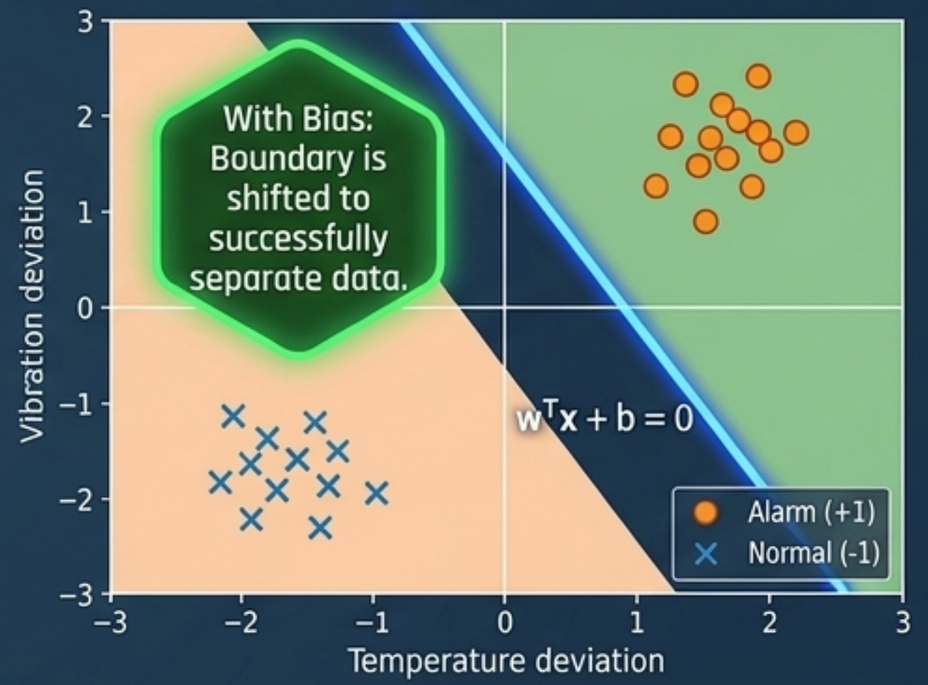
If s <= 0, output is -1

THE ROLE OF BIAS IN LINEAR SEPARATION

WITHOUT BIAS (b = 0)



WITH BIAS (b ≠ 0)



KEY GEOMETRIC RULES

1. Changing **w** changes the direction of the boundary (**w** is always perpendicular to the line).
2. Changing **b** shifts the boundary's location in space.

DIAGNOSTIC MATRIX

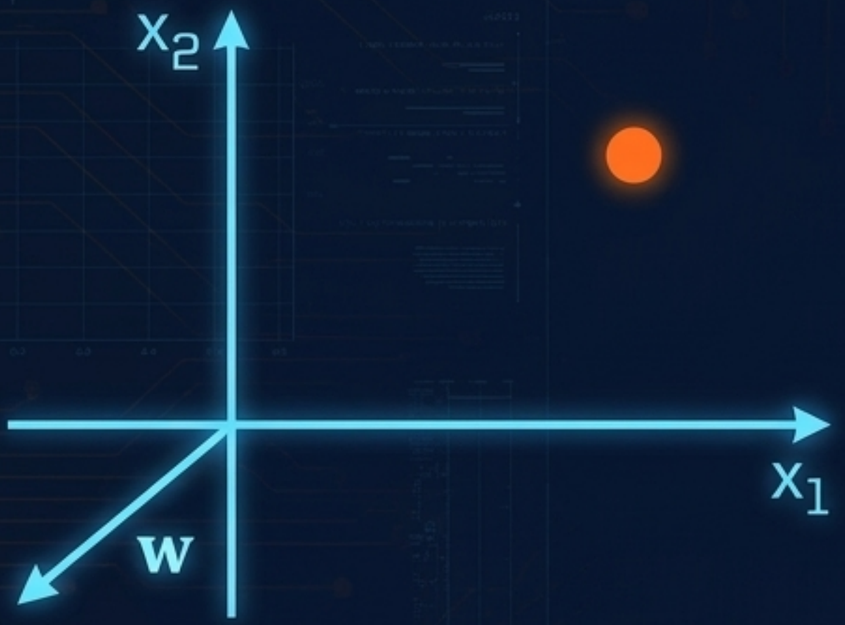
	Actual Class: $y_i = +1$	Actual Class: $y_i = -1$
Predicted Score Sign: $s > 0$	Positive x Positive = > 0 (Correct)	Positive x Negative = <= 0 (Mistake detected!)
Predicted Score Sign: $s < 0$	Negative x Positive = <= 0 (Mistake detected!)	Negative x Negative = > 0 (Correct)

The Signed Margin Detector

$$y_i(w^T x_i + b) \leq 0$$

PERCEPTRON UPDATE: THE PIVOT RULE

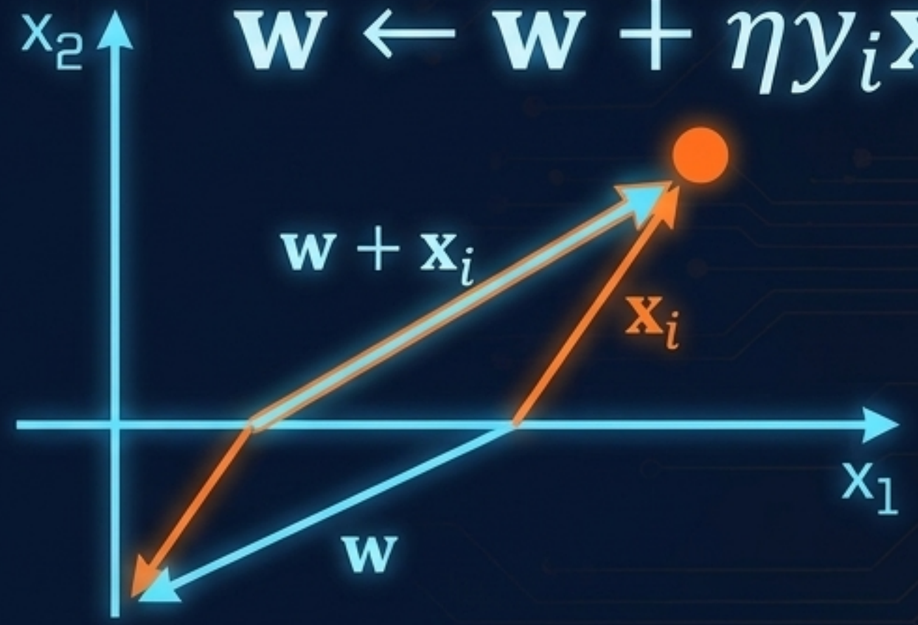
PANEL 1: THE ERROR



Weight vector $\mathbf{w}$ points away from positive example.

PANEL 2: THE CORRECTION

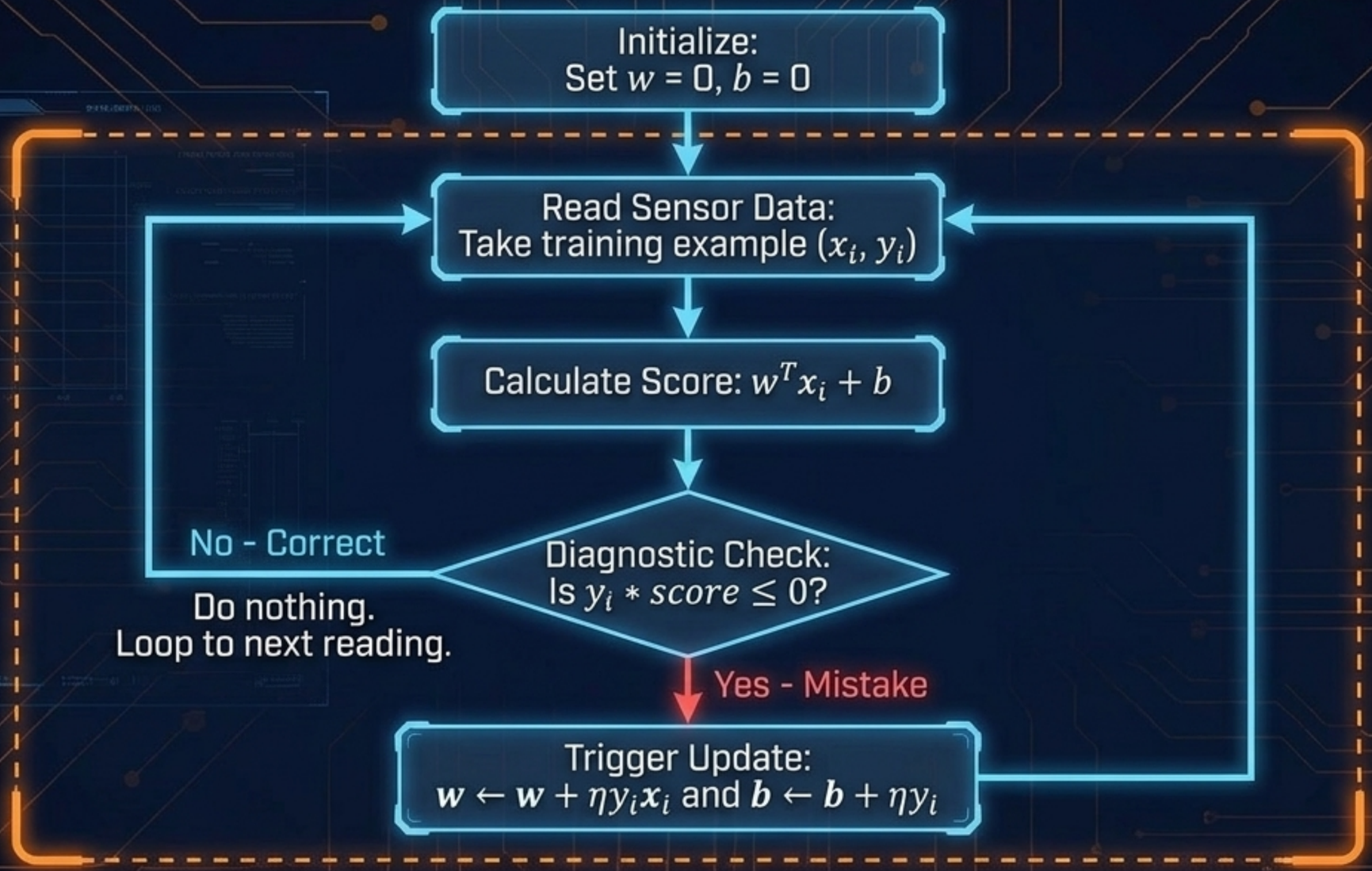
$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$$



New resulting vector $\mathbf{w} + \mathbf{x\_i}$ pivoting directly toward the example.

THE UPDATE RULE FORCES THE WEIGHT VECTOR TO PIVOT TOWARD MISSED POSITIVE EXAMPLES (AND AWAY FROM MISSED NEGATIVE EXAMPLES), INCREASING THE SCORE FOR THAT ITEM NEXT TIME.

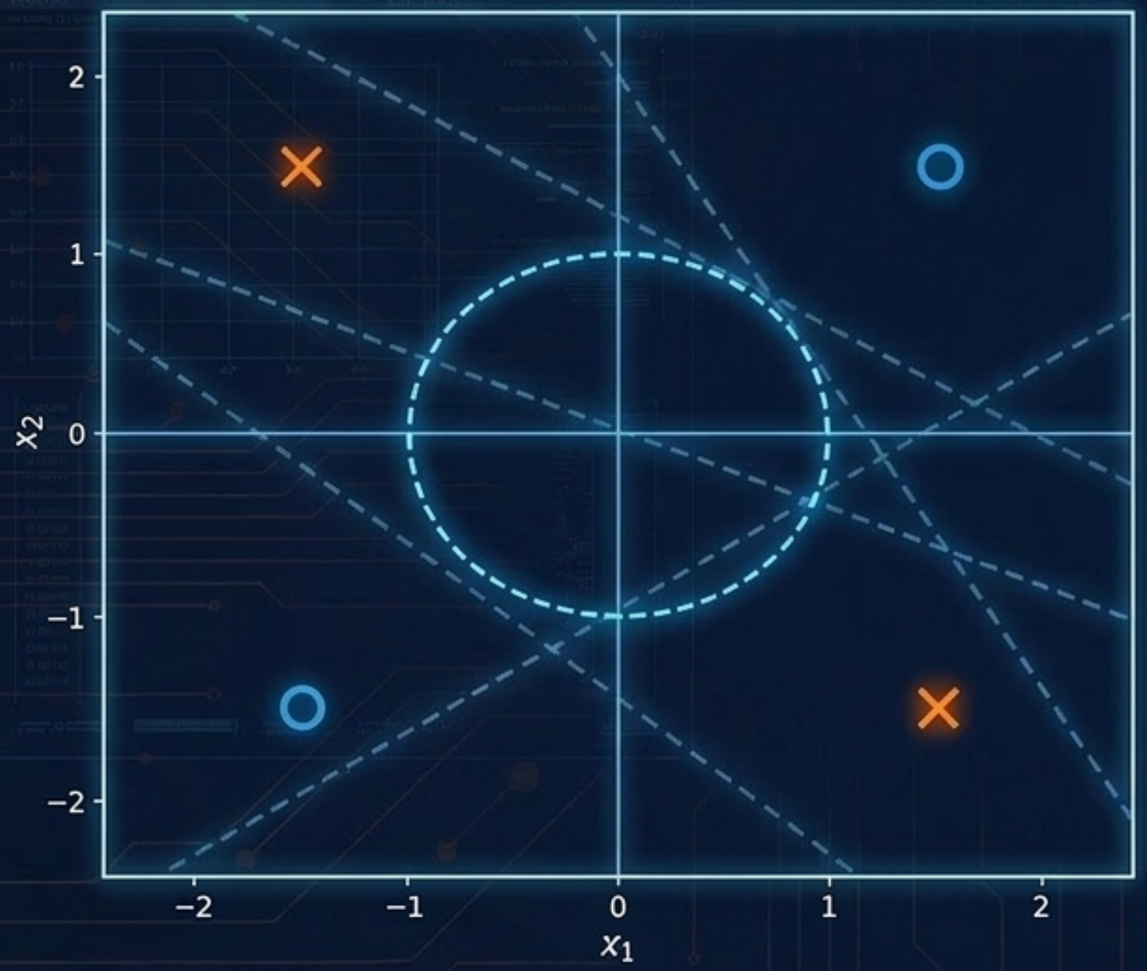
PERCEPTRON LEARNING ALGORITHM: THE CLOSED-LOOP CYCLE



Repeat for several Epochs until linearly separated (no mistakes).

XOR PATTERN: LINEARITY LIMITATION DETECTED

XOR SCATTER PLOT & BOUNDARY FAILURE



DIAGNOSTIC READOUT

SYSTEM FAILURE DETECTED

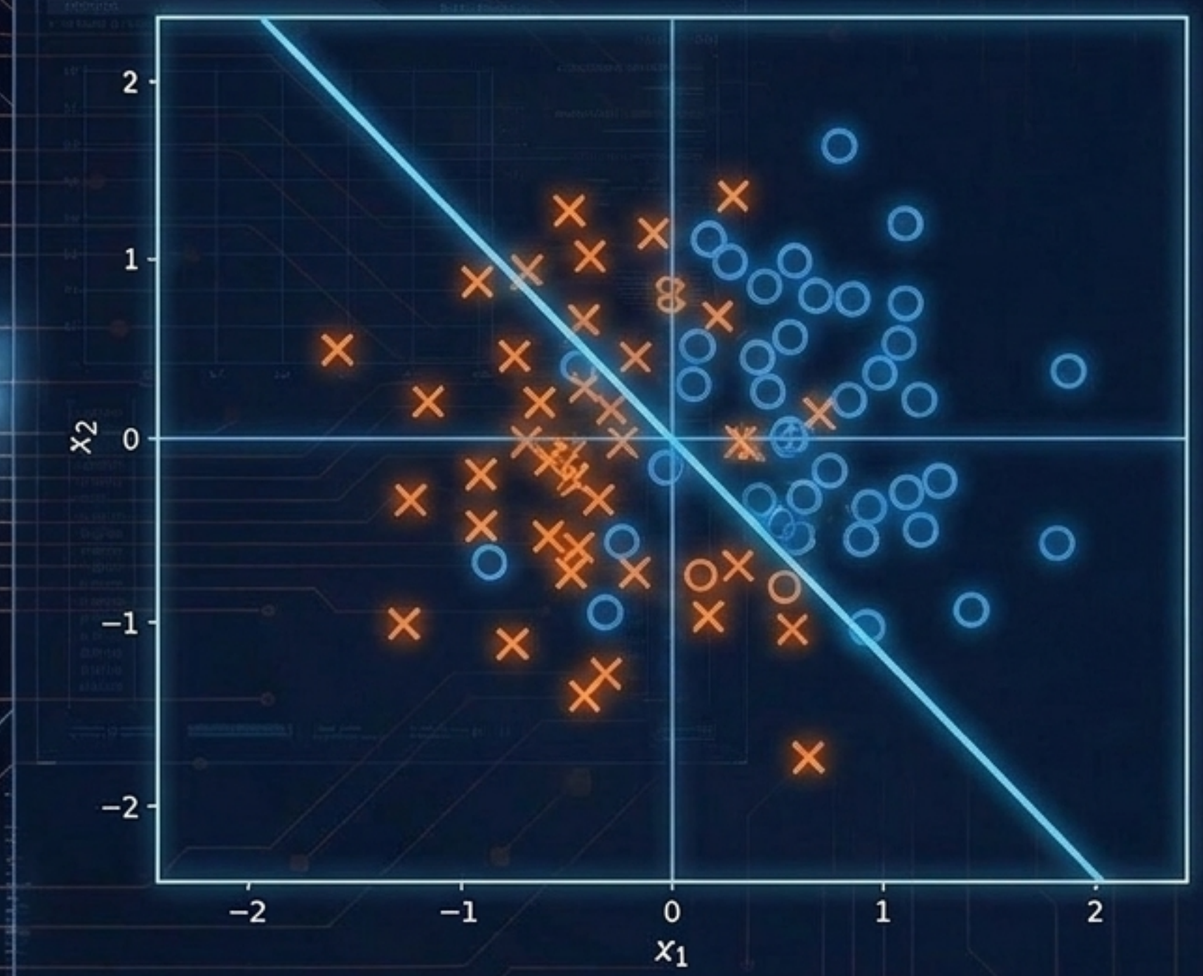
Limitation 1: Cannot learn a nonlinear decision boundary.

Reason: The equation $w^T x + b = 0$ strictly defines a flat line or hyperplane.

Conclusion: A single Perceptron cannot bend.

TELEMETRY SCATTER PLOT & BOUNDARY FAILURE

TELEMETRY SCATTER PLOT & BOUNDARY FAILURE



DIAGNOSTIC READOUT

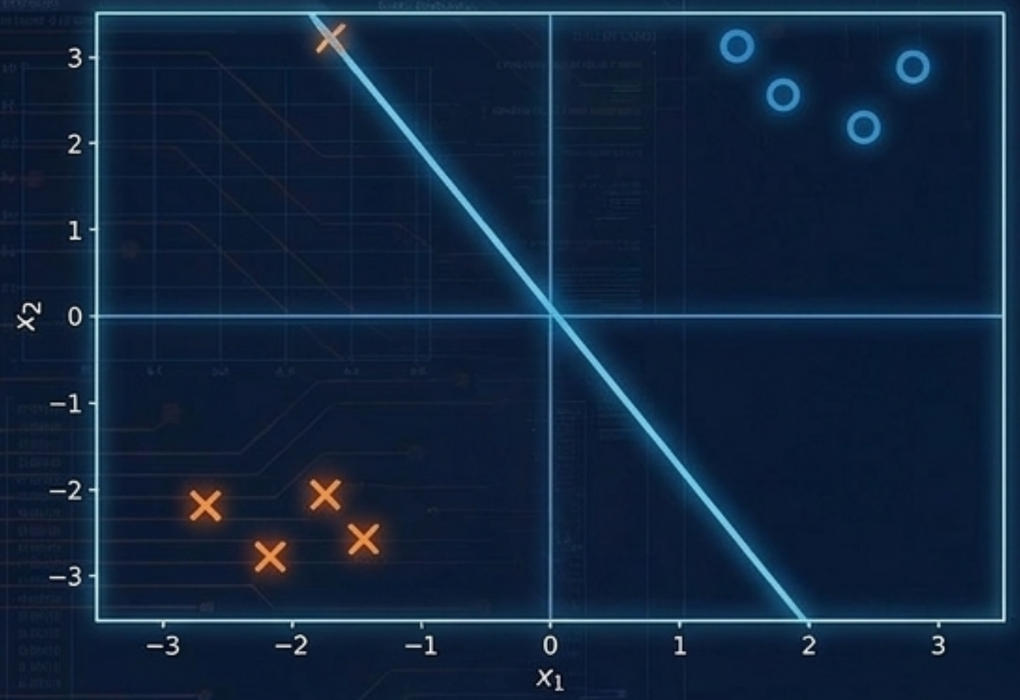
Limitation 2: May not converge on non-separable data.

Result: If perfect separation is impossible due to sensor noise or missing features, the Perceptron loop will run infinitely, constantly shifting weights without ever reaching zero error.

Engineering Fix: Practical implementations require an artificial ‘maximum epochs’ stopping condition.

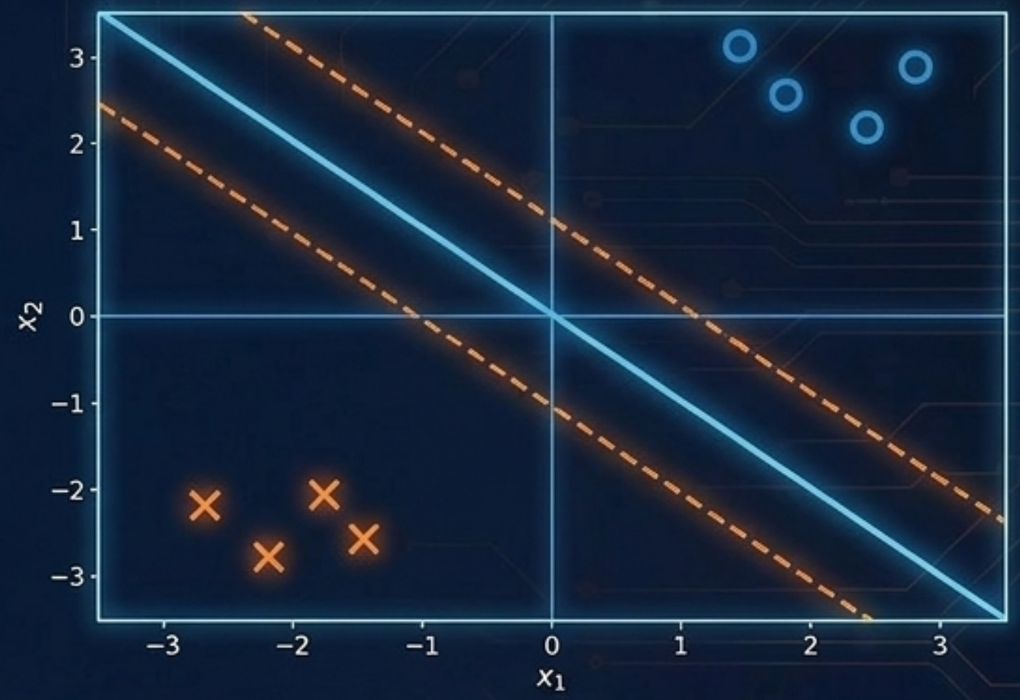
THE PERCEPTRON: TECHNICALLY CORRECT

THE PERCEPTRON: TECHNICALLY CORRECT



Technically correct, highly unstable.

MAXIMUM MARGIN: OPTIMAL STABILITY



Optimal stability.

DIAGNOSTIC READOUT

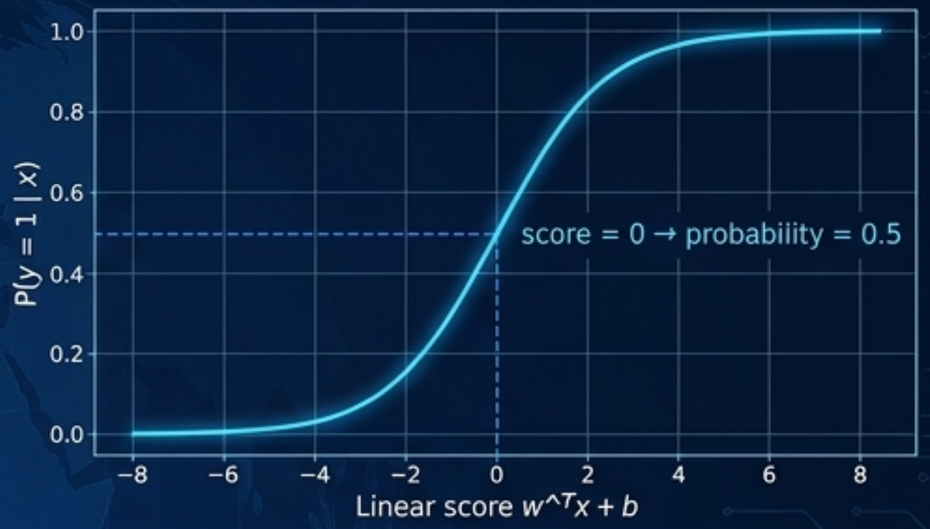
Limitation 3: Does not find the **best** boundary. It stops at the very first line that works, making it highly sensitive to training order and initial weights.

THE PROBABILITY ILLUSION

s = 0.01

s = 100

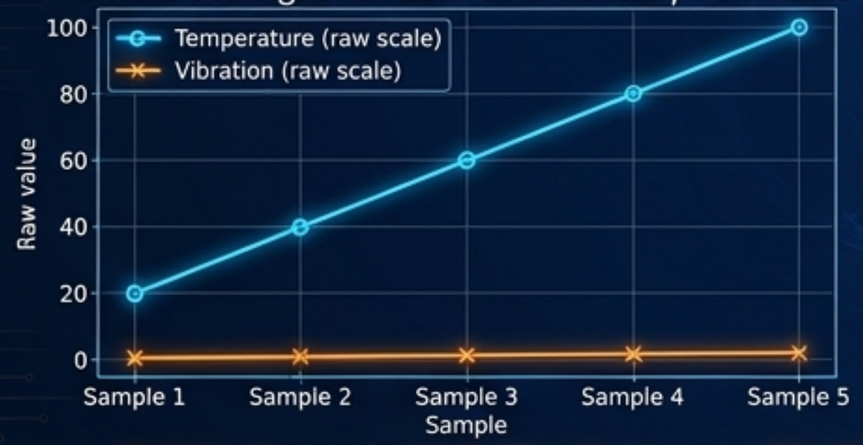
Output: +1



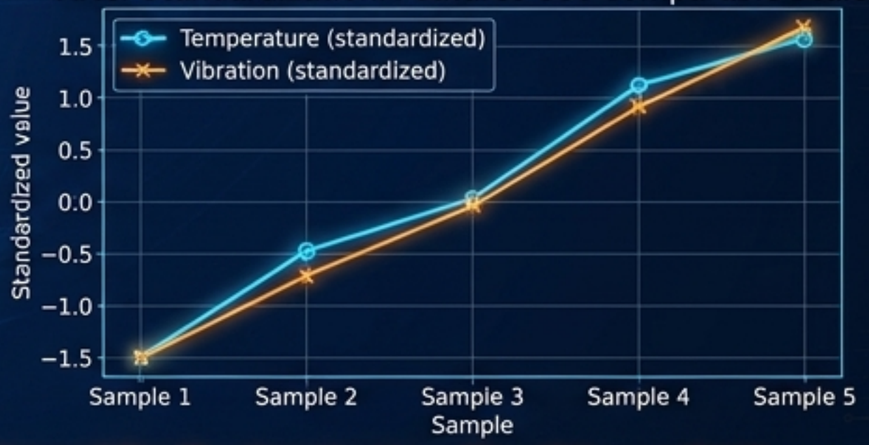
Scores are NOT probabilities.

THE SCALE THREAT

Before Scaling: One Feature Numerically Dominates



After Standardization: Features Use Comparable Scales



If data is unscaled, the feature with the largest raw numbers blindly dominates the dot product.

THE PERCEPTRON'S LIMITATIONS & MODERN PATCHES

Hardware Flaw (Perceptron Limitation)	Modern Patch (Related Improvement)
Cannot bend (Nonlinear data)	Kernel Methods & Neural Networks
Infinite loops on messy data	Soft-margin SVM & Logistic Regression
Settles for unstable boundaries	Maximum-margin SVM
Outputs binary, not probabilities	Logistic Regression
Blindly dominated by large numbers	Feature Standardization

Key Insight: Every modern linear model was designed specifically to patch one of these exact structural flaws.

THE TRANSLATION: SCIKIT-LEARN ROSETTA STONE

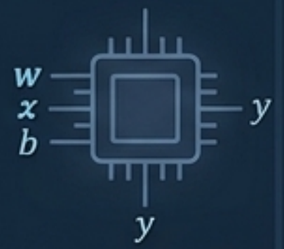
Theoretical Math	Python Code
Update rule loop until convergence	`model.fit(X, y)`
$\hat{y} = sign(w^T x + b)$	`model.predict(X)`
Limit infinite loops on noise	`max_iter=1000`
Percentage of $y_i(w^T x_i + b) > 0$	`model.score(X, y)`

Scikit-Learn abstracts the vector geometry into four simple commands.

PERCEPTRON BLUEPRINT: ENGINEERING SCHEMATIC

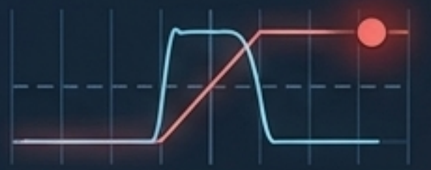
1. THE ENGINE

$$\hat{y} = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$



2. THE DIAGNOSTIC

Mistake if $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \leq 0$



3. THE CORRECTION PIVOT

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$$
$$\boldsymbol{b} \leftarrow b + \eta y_i$$

4. THE GOLDEN RULE

Will always converge on linearly separable data.
Will always fail on nonlinear data.