

Mathematics for Machine Learning — Student Cheat Sheet

Instructor: Pushpjeet Cholkar · Level: intermediate beginner · Needs: high-school maths + basic Python/NumPy
Full tutorial: pushpjeet.com/tutorials/mathematics-for-machine-learning/ · Lab: pushpjeet.com/downloads/maths-for-ml-lab.zip

Use this as a one-sitting review before the lab or an interview. Numbers match the lab run on Python 3.13.5 / NumPy 2.5.3.

Symbols in 60 seconds

Symbol	Say it	Meaning in ML
$\mathbf{x}$	"vector x"	Features for one example
$\mathbf{w}, \theta$	"weights" / "theta"	Parameters you learn
b	"bias"	Offset term
$\hat{y}$	"y-hat"	Prediction
$J(\mathbf{w})$	"cost J of w"	Scalar loss / cost
$\frac{dJ}{dw}$	"dJ dw"	Derivative (1D slope)
$\frac{\partial J}{\partial w_i}$	"partial J partial w_i"	Slope when only w_i moves
∇J	"grad J" / "nabla J"	Vector of all partials
η	"eta"	Learning rate
$\ \mathbf{a}\ _2$	"L2 norm of a"	Euclidean length
$\mathbf{a} \cdot \mathbf{b}$	"a dot b"	Sum of $a_i b_i$
$\mathbf{X}^\top$	"X transpose"	Rows $\leftrightarrow$ columns

Must-know equations

Linear prediction

$$\hat{y} = \mathbf{w} \cdot \mathbf{x} + b = \sum_i w_i x_i + b$$

L2 norm / distance

$$\|\mathbf{a}\|_2 = \sqrt{\sum_i a_i^2}, \quad d(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2$$

Cosine similarity

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2}$$

Gradient descent update

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla J(\mathbf{w})$$

MSE (batch)

$$J(\theta) = \frac{1}{n} \|\mathbf{X}\theta - \mathbf{y}\|_2^2, \quad \nabla_{\theta} J = \frac{2}{n} \mathbf{X}^\top (\mathbf{X}\theta - \mathbf{y})$$

Confusions to kill early

Pair	Remember
Scalar vs vector	Loss J is a scalar; features/weights are vectors
Derivative vs gradient	1 variable $\rightarrow$ derivative; many $\rightarrow$ gradient (a vector)
Dot vs <code>*</code> in NumPy	Dot $\rightarrow$ one number; <code>*</code> $\rightarrow$ element-wise array
Subtract vs add gradient	Subtract to minimise cost
L1 vs L2	L1 = sum of abs; L2 = sqrt of sum of squares

Lab checkpoints (expected numbers)

Run: `python maths_for_ml_lab.py` (from the [lab ZIP](#))

Check	Expected
$8 \cdot 4 + 3 \cdot 7 + 20$	73.0
Batch preds	[73, 83, 60]
$\ (3, 4)\ _2$	5.0
Cosine of parallel vectors	1.0000
1D GD final w (25 steps, $\eta = 0.1$)	≈ 2.981
2D GD final	$\approx [1.970, -1.000]$
Linear reg recovered θ	$\approx [4.99, 2.00, 0.93]$

Danger demo: $\eta = 1.1$ on $J = (w - 3)^2$ from $w = -2 \rightarrow$ cost climbs (25, 36, 52, ...).

5-line NumPy patterns

```
import numpy as np
y_hat = X @ w + b           # batch predictions
loss = np.mean((y_hat - y) ** 2) # MSE
grad = (2 / n) * (X.T @ (y_hat - y)) # MSE gradient w.r.t. w (X includes bias col)
w = w - lr * grad           # descent step
np.linalg.norm(a, ord=2)     # L2 norm
```

Interview pocket answers

- Gradient vs derivative:** gradient = all partials as a vector; derivative = 1D case.
- Why minus ∇J ?** Gradient points uphill on cost; we go downhill.
- Too-large η :** overshoot / diverge — loss can increase.
- Bias trick:** append a column of ones; bias becomes one more weight.
- Cosine vs dot:** cosine = normalised dot; equal to dot on unit vectors (RAG retrieval).

Next pages on pushpjeet.com

- Pillar: </tutorials/mathematics-for-machine-learning/>
- Foundations: </tutorials/ai-ml-fundamentals/>
- Apply vectors to retrieval: </tutorials/rag-tutorial-python/>