Understanding the Core of Learning with Complete Derivations

Gradient, Regularization, and Sparsity

Data Science Expert

May 11, 2026

Contents

1 Part 1: What is a Gradient and Why Do We Use It? 4
1.1 The Core Definition . . . 4
1.2 Mathematical Definition . . . 4
1.3 The Three Critical Questions Gradient Answers . . . 4
1.3.1 Question 1: Which direction should I move? . . . 4
1.3.2 Question 2: How fast should I move? (Step size / Learning Rate) . . . 4
1.3.3 Question 3: Which variable matters most? . . . 5
1.4 What Happens WITHOUT Gradient? . . . 5
1.5 Why Gradient is Essential . . . 5

2 Part 2: How Regularization Connects to Gradient 5
2.1 The Big Picture . . . 5
2.2 Mathematical Connection . . . 6
2.3 What Regularization Does to the Gradient . . . 6

3 Part 3: CORRECTING A MAJOR MISCONCEPTION 6
3.1 Important Correction! . . . 6
3.2 Correct Statement . . . 7

4 Part 4: Why L1 Pushes Weights Exactly to Zero 7
4.1 Intuitive Explanation . . . 7
4.2 Mathematical Derivation . . . 7
4.2.1 For L1 (Lasso) . . . 7
4.3 Geometric Intuition . . . 7
4.4 Detailed Example: L1 Pushes to Zero . . . 8

5 Part 5: Why L2 Shrinks Weights But Never to Zero 8
5.1 Intuitive Explanation . . . 8
5.2 Mathematical Derivation . . . 8
5.3 Geometric Intuition . . . 8
5.4 Comparison Table . . . 9
5.5 Direct Numerical Example . . . 9

6 Part 6: Without Gradient, Regularization Wouldn't Work 9

7 Part 7: Summary and Final Thoughts **10**
7.1 The Complete Picture 10
7.2 Visual Summary: L1 vs L2 Geometry 10
7.3 Final Deep Concluding Thought 10

1 Part 1: What is a Gradient and Why Do We Use It?

1.1 The Core Definition

Think of a gradient as a **direction + strength signal** that tells you:

"If you want to improve something (reduce loss), move this way, and this much."

1.2 Mathematical Definition

For a function $f(\mathbf{w})$ with multiple variables, the gradient is a vector of partial derivatives:

$$\nabla f(\mathbf{w}) = \left[\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \ldots, \frac{\partial f}{\partial w_n}\right]^T$$

What this tells us:

- **Direction:** Gradient points toward the **fastest increase**
- **Opposite direction:** $-\nabla f$ points toward the **fastest decrease**
- **Magnitude:** How steep the slope is

1.3 The Three Critical Questions Gradient Answers

1.3.1 Question 1: Which direction should I move?

Why this matters: Without direction, you're just wandering randomly.

In high-dimensional spaces (millions of parameters), random movement is useless. The gradient gives the **correct direction** for fastest improvement.

Update rule:

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \nabla L(\mathbf{w}_{\text{old}})$$

where η is the learning rate.

What happens if we don't use gradient?

- Random search in 1,000,000 dimensions is impossible
- You would never converge
- Loss would oscillate or increase

1.3.2 Question 2: How fast should I move? (Step size / Learning Rate)

The trade-off:

Step Size	**Effect**	**Problem**
Too large (η big)	Overshoots minimum	Training unstable, diverges
Too small (η tiny)	Crawls slowly	Takes forever, may get stuck
Just right	Stable convergence	Optimal efficiency

Why learning rate matters:

$$\text{If } \eta > \frac{2}{\lambda_{\max}}, \text{ gradient descent diverges}$$

where $\lambda_{\max}$ is the largest eigenvalue of the Hessian.

1.3.3 Question 3: Which variable matters most?

The gradient is a vector; each component $\frac{\partial L}{\partial w_j}$ tells how sensitive the loss is to that specific weight.

Why this matters:

- Not all parameters are equally important
- Important features get large gradients → large updates
- Irrelevant features get small gradients → tiny updates
- This is how the model learns which weights are crucial

1.4 What Happens WITHOUT Gradient?

Without Gradient

- **No direction:** You would guess random weight updates
- **No speed:** No way to know step size
- **No importance:** All weights updated equally
- **Result:** Training never converges; random search in high dimensions fails

1.5 Why Gradient is Essential

With Gradient

- **Move intelligently:** Follow steepest descent
- **Move efficiently:** Use learning rate schedules
- **Focus on what matters:** Large gradients for important weights
- **Result:** Converges to good solutions in minutes/hours

2 Part 2: How Regularization Connects to Gradient

2.1 The Big Picture

- **Gradient:** Tells you how to update parameters to reduce loss

- **Regularization:** Modifies the loss so the model doesn't overfit
- **Connection:** When you change the loss, you automatically change its gradient

2.2 Mathematical Connection

We don't minimize just the data loss anymore. We add a penalty:

$$L_{\text{total}}(\mathbf{w}) = L_{\text{data}}(\mathbf{w}) + \lambda R(\mathbf{w})$$

Taking the gradient:

$$\nabla L_{\text{total}}(\mathbf{w}) = \nabla L_{\text{data}}(\mathbf{w}) + \lambda \nabla R(\mathbf{w})$$

The update rule becomes:

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \left(\nabla L_{\text{data}}(\mathbf{w}_{\text{old}}) + \lambda \nabla R(\mathbf{w}_{\text{old}}) \right)$$

Interpretation:

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \nabla L_{\text{data}}(\mathbf{w}_{\text{old}}) - \eta \lambda \nabla R(\mathbf{w}_{\text{old}})$$

- First term: Original learning signal
- Second term: Regularization "nudge" pulling weights toward simplicity

2.3 What Regularization Does to the Gradient

Regularization nudges the gradient so that:

- Weights don't grow too large
- Model stays simple
- Overfitting is reduced

3 Part 3: CORRECTING A MAJOR MISCONCEPTION

3.1 Important Correction!

WARNING: Common Mistake

What you heard was INCORRECT.

- **L2 does NOT increase sparsity.** It does the opposite.
- **L1 creates sparsity** (pushes weights to zero)
- **L2 shrinks weights** (makes them small but non-zero)

3.2 Correct Statement

4 Part 4: Why L1 Pushes Weights Exactly to Zero

4.1 Intuitive Explanation

Imagine you have a tax on your possessions:

- **L1 tax:** You pay \$1 for every dollar of wealth you have. If you have \$1, you pay \$1 (keep \$0). If you have \$0, you pay \$0. The tax is so high that you prefer to have zero wealth.
- **L2 tax:** You pay a percentage of your wealth. You can never reach zero because you pay a fraction, not a fixed amount.

4.2 Mathematical Derivation

4.2.1 For L1 (Lasso)

Penalty: $R(w) = |w|$

The subgradient (for $w \neq 0$):

$$\frac{\partial R}{\partial w} = \text{sign}(w) = \begin{cases} +1 & \text{if } w > 0 \\ -1 & \text{if } w < 0 \end{cases}$$

The update rule for a single weight:

$$w_{\text{new}} = w_{\text{old}} - \eta \left(\frac{\partial L_{\text{data}}}{\partial w} + \lambda \cdot \text{sign}(w) \right)$$

Why weights become exactly zero: If w is small positive, $\text{sign}(w) = +1$, so:

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \lambda - \eta \frac{\partial L_{\text{data}}}{\partial w}$$

The constant pull $-\eta\lambda$ pushes w toward zero. If w_{old} is small enough, it crosses zero in one step. Once at zero, the subgradient allows it to stay at zero.

4.3 Geometric Intuition

L1 Geometry

L1 constraint: $\|w\|_1 \leq t$ is a **diamond** shape. The corners of the diamond lie on the axes (where some $w_i = 0$). When the unconstrained solution lies outside the diamond, the constrained solution is at a **corner** $\rightarrow$ sparsity!

4.4 Detailed Example: L1 Pushes to Zero

Suppose $w = 0.01$, $\lambda = 0.1$, $\eta = 0.01$, and assume $\frac{\partial L_{\text{data}}}{\partial w} \approx 0$ near zero.

L1 update:

$$w_{\text{new}} = 0.01 - 0.01 \times 0.1 \times (+1) = 0.01 - 0.001 = 0.009$$

Next step: $0.009 - 0.001 = 0.008$. Continue until w crosses zero. Once negative, $\text{sign}(w) = -1$ pushes it back up. It oscillates and can become exactly zero due to subgradient.

5 Part 5: Why L2 Shrinks Weights But Never to Zero

5.1 Intuitive Explanation

L2 penalty is like a **spring** attached to each weight. The spring pulls the weight toward zero, but the pull gets **weaker** as the weight gets smaller. It never pulls hard enough to make the weight exactly zero.

5.2 Mathematical Derivation

For L2 (Ridge): $R(w) = w^2$

The gradient:

$$\frac{\partial R}{\partial w} = 2w$$

The update rule:

$$w_{\text{new}} = w_{\text{old}} - \eta \left(\frac{\partial L_{\text{data}}}{\partial w} + 2\lambda w_{\text{old}} \right)$$

Why weights never become exactly zero: Write as:

$$w_{\text{new}} = w_{\text{old}}(1 - 2\eta\lambda) - \eta \frac{\partial L_{\text{data}}}{\partial w}$$

If w_{old} is very small (say 10^{-10}), then $1 - 2\eta\lambda \approx 1$ for typical $\eta\lambda$. The update is tiny. To reach exactly zero, you would need infinite steps. In practice, weights approach zero **asymptotically** but never reach it.

5.3 Geometric Intuition

L2 Geometry

L2 constraint: $\|w\|_2^2 \leq t$ is a **circle**. Circles have no corners. The constrained solution can touch the circle at any point, but never at an axis unless the unconstrained solution is already there. So weights are shrunk but rarely become exactly zero.

5.4 Comparison Table

Property	**L1 (Lasso)**	**L2 (Ridge)**
Penalty	$\lambda\|w\|$	λw^2
Gradient	$\lambda \cdot \text{sign}(w)$	$2\lambda w$
Update effect	Constant pull toward zero	Pull proportional to w
Small w behavior	Still pushed with full force	Pull becomes very weak
Sparsity	Yes (weights exactly zero)	No (weights small but non-zero)
Feature selection	Yes	No
Solution uniqueness	Not always unique	Always unique
Geometric shape	Diamond (corners on axes)	Circle (no corners)

5.5 Direct Numerical Example

Comparing L1 vs L2

Suppose $w = 0.5$, learning rate $\eta = 0.1$, regularization strength $\lambda = 0.1$, and assume data gradient $\frac{\partial L_{\text{data}}}{\partial w} = 0$ for simplicity.
L1 update:

$$w_{\text{new}} = 0.5 - 0.1 \times 0.1 \times (+1) = 0.5 - 0.01 = 0.49$$

Each step subtracts exactly 0.01. After 50 steps, $w = 0$.
L2 update:

$$w_{\text{new}} = 0.5 - 0.1 \times 0.1 \times (2 \times 0.5) = 0.5 - 0.1 \times 0.1 \times 1 = 0.5 - 0.01 = 0.49$$

But next step: $w = 0.49$, $2w = 0.98$, subtract $0.1 \times 0.1 \times 0.98 = 0.0098$, so $w = 0.4802$. The step size decreases as w decreases. It asymptotically approaches zero but never reaches it.

6 Part 6: Without Gradient, Regularization Wouldn't Work

Why Regularization Needs Gradient

You could define a penalty, but:

- Without gradient, you don't know how to apply the penalty
- Without direction, you can't "pull weights back" intelligently
- The regularization term would be just a number, not a force

Conclusion: Regularization **relies on the gradient** to enforce its constraints.

7 Part 7: Summary and Final Thoughts

7.1 The Complete Picture

Key Takeaways

1. **Gradient** gives direction, speed, and importance — three essential answers for learning.
2. **Without gradient**, you're randomly guessing in high-dimensional space. Training never converges.
3. **Regularization** modifies the loss, which changes the gradient.
4. **L1 regularization** creates sparsity (weights exactly zero) because its gradient is constant ($\pm\lambda$), constantly pushing toward zero.
5. **L2 regularization** shrinks weights (makes them small but non-zero) because its gradient is proportional to w, weakening as w gets small.
6. The statement "L2 increases sparsity and L1 decreases sparsity" is **backwards**. L1 creates sparsity; L2 does not.

7.2 Visual Summary: L1 vs L2 Geometry

L1 (Lasso) - Diamond	**L2 (Ridge) - Circle**
Corners on axes → sparsity	No corners → no sparsity
Constraint: $\|w_1\| + \|w_2\| \leq t$	Constraint: $w_1^2 + w_2^2 \leq t$
Solution at corner → some weights zero	Solution on curve → no weights zero

7.3 Final Deep Concluding Thought

"The gradient is the compass of machine learning. Regularization is the map that adds guardrails. Without the compass, you wander aimlessly. Without the guardrails, you fall off the cliff of overfitting. Together, they guide you to the valley of generalization."