Regularization in MLPs

Atharv Bhatt

January 18, 2026

Contents

1 Introduction **2**

2 Overfitting and Underfitting **2**

3 Regularization: Concept **2**
3.1 What is Regularization? 2

4 Explicit Regularization **3**
4.1 L2 Regularization (Ridge / Weight Decay) 3
4.2 L1 Regularization (Lasso) 3
4.3 Elastic Net 3

5 Implicit Regularization **3**
5.1 Types of Implicit Regularization 4

6 Other Regularization Techniques **4**
6.1 Weight Decay 4
6.2 Dropout 4
6.3 BatchNorm / LayerNorm 4
6.4 Early Stopping 4

7 Summary Table of Techniques **4**

8 Practical Intuition **5**

9 Conclusion **5**

1 Introduction

In machine learning, particularly in **Multilayer Perceptrons (MLPs)**, a primary challenge is to ensure that the model **generalizes well** on unseen data. While deep models are powerful and can approximate complex functions, they are prone to **overfitting**, especially on small or noisy datasets.

Regularization is the set of techniques aimed at **reducing overfitting**, controlling model complexity, and improving generalization. It balances the **bias-variance trade-off**, allowing models to learn meaningful patterns while ignoring noise.

2 Overfitting and Underfitting

Before diving into regularization methods, it is crucial to understand the **problems regularization solves**.

Concept	**Description**	**Training Error**	**Validation Error**
Underfitting	Model too simple to capture patterns	High	High
Overfitting	Model too complex, memorizes training data	Low	High
Good Fit	Model captures true patterns, generalizes well	Low	Low

Intuition:

- **Underfitting:** "Too weak" $\rightarrow$ can't even learn the training data.
- **Overfitting:** "Too strong" $\rightarrow$ learns training data perfectly, including noise, fails on new data.

3 Regularization: Concept

3.1 What is Regularization?

Regularization refers to techniques that **prevent overfitting by controlling model complexity**.

$$L_{\text{total}} = L_{\text{data}} + \lambda \cdot \text{Penalty}$$

Where:

- L_{data}: original loss (MSE, cross-entropy, etc.)
- Penalty: measures model complexity (weights, activations, etc.)
- λ: regularization strength

Key Idea: Reduces model sensitivity to noise (reduces variance) while slightly increasing bias (model may not perfectly fit training data), resulting in better generalization on unseen data.

4 Explicit Regularization

Explicit regularization **adds a penalty directly to the loss function**.

4.1 L2 Regularization (Ridge / Weight Decay)

Idea: Penalizes large weights, encourages smooth, small-magnitude weights.

$$L_{\text{total}} = L_{\text{data}} + \lambda \sum w_i^2$$

Effect: Shrinks weights, rarely zero → reduces variance, improves generalization

$$\frac{\partial L}{\partial w_i} = \frac{\partial L_{\text{data}}}{\partial w_i} + 2\lambda w_i$$

Usage: Default in MLPs; works well with correlated features

4.2 L1 Regularization (Lasso)

Idea: Penalizes absolute weights → promotes sparsity, automatic feature selection

$$L_{\text{total}} = L_{\text{data}} + \lambda \sum |w_i|$$

Effect: Some weights exactly zero → ignores irrelevant features
When to use: High-dimensional data, interpretability needed

4.3 Elastic Net

Idea: Combines L1 and L2 → sparse yet stable solutions

$$L_{\text{total}} = L_{\text{data}} + \lambda_1 \sum |w_i| + \lambda_2 \sum w_i^2$$

Effect: Benefits from both sparsity (L1) and smoothness (L2)
When to use: Correlated, high-dimensional features

5 Implicit Regularization

Implicit regularization **does not explicitly add penalties to the loss**, but the **training method or architecture naturally prevents overfitting**.

5.1 Types of Implicit Regularization

Technique	How it works	Effect
Early Stopping	Stop training when validation loss stops improving	Prevents memorizing noise
Dropout	Randomly disables neurons during training	Reduces co-adaptation, adds stochasticity
BatchNorm / LayerNorm	Normalizes activations	Adds noise, stabilizes learning
SGD / Mini-batch training	Noise in gradient updates	Prevents convergence to sharp minima, reduces variance
Data Augmentation	Input transformations	Model learns invariant patterns

Intuition: "You didn't tell the model to be simple, but the way it is trained naturally keeps it simple."

—

6 Other Regularization Techniques

6.1 Weight Decay

- Essentially L2 regularization in deep learning - Directly shrinks weights during optimization

6.2 Dropout

- Randomly drop hidden neurons during training - Forces network to learn redundant representations → robust features - Typically: $p = 0.3 - 0.5$ in hidden layers

6.3 BatchNorm / LayerNorm

- BatchNorm: normalizes activations per batch - LayerNorm: normalizes activations per layer - Adds small stochastic noise → acts as implicit regularizer - Stabilizes training, can sometimes replace dropout

6.4 Early Stopping

- Stop training when validation loss stops improving - Avoids overfitting without modifying loss - Slightly increases bias, dramatically reduces variance

—

7 Summary Table of Techniques

Technique	Type	Bias	Variance	Sparsity	Notes
L1	Explicit	↑	↓	✓	Feature selection
L2	Explicit	↑	↓	×	Default in deep networks
ElasticNet	Explicit	↑	↓	✓	Sparse + stable
Dropout	Implicit	↑	↓	×	Deep MLPs, stochastic noise
BatchNorm	Implicit	↑	↓	×	Stabilizes training
LayerNorm	Implicit	↑	↓	×	Small batch / sequential data
Early Stopping	Implicit	↑	↓	×	Monitor validation loss

—

8 Practical Intuition

- **Overfitting → high train, low validation** → use regularization - **Underfitting → low train, low validation** → reduce regularization / increase model capacity - Regularization reduces variance, slightly increases bias → improves generalization - Explicit regularization: add penalty to loss - Implicit regularization: occurs naturally through training/architecture

Mnemonic: "Explicit = I told you to be simple. Implicit = you naturally behave simply because of how you are trained."

—

9 Conclusion

Regularization in MLPs is **essential for good generalization**. By combining **explicit methods (L1, L2, ElasticNet)** and **implicit methods (Dropout, BatchNorm, Early Stopping, SGD noise)**, we can:

- Prevent overfitting
- Stabilize training
- Improve generalization
- Ensure robust models even with limited or noisy data

Practical recommendation: - Default deep learning MLP → L2 + Dropout + Early Stopping - High-dimensional tabular data → L1 or ElasticNet

Regularization is **about guiding the model to learn patterns, not memorize noise**.