Regularization to Prevent Overfitting -AliHaghayeghi

Introduction

Regularization refers to **any modification of a learning algorithm that reduces generalization error without necessarily reducing training error**. Its purpose is to **prevent overfitting**, enabling models to generalize from limited, noisy, or imperfect training data.

Formally, given a function f_θ with parameters θ, trained to minimize empirical risk:

$$\hat{R}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i),$$

a regularized objective modifies this to:

$$\hat{R}_{\text{reg}}(\theta) = \hat{R}(\theta) + \lambda\Omega(\theta),$$

where $\Omega(\theta)$ encodes prior assumptions such as smoothness, sparsity, or simplicity.

---

2. Foundations of Overfitting

2.1 Generalization and Bias–Variance Tradeoff

- **Underfitting** → high bias, low variance
- **Overfitting** → low bias, high variance
- **Regularization redistributes bias and variance** to optimal generalization.

Textual Diagram: Bias–Variance Tradeoff Curve

```
Error
│      * High variance region
│     *  (overfitting)
│    *
│   *
│  * Low variance region
│ *  (underfitting)
└──────────────────────────────→ Model Complexity
```

2.2 Hypothesis Complexity Measures

Different theoretical tools relate complexity to generalization:

- **VC dimension**
- **Rademacher complexity**
- **Description length / MDL**
- **Norm-based bounds (e.g., $||W||_2$)**

Many regularizers explicitly or implicitly reduce one of these complexity measures.

---

3. Classical Regularization Methods

3.1 L2 Regularization (Weight Decay)

Objective:

$$\hat{R}_{L2} = \hat{R}(\theta) + \frac{\lambda}{2} \| \theta \|_2^2$$

Effect:

- Shrinks weights smoothly toward zero.
- Encourages distributed, non-sparse representations.
- Equivalent to a Gaussian prior in Bayesian interpretation.

L2 regularization is often implemented as **decoupled weight decay** (AdamW):

$$\theta \leftarrow \theta - \eta(\nabla_\theta \hat{R} + \lambda\theta).$$

Why decoupling matters: Adam's adaptive scaling interacts poorly with naïve L2, leading to suboptimal regularization magnitude.

3.2 L1 Regularization (Sparsity)

$$\hat{R}_{L1} = \hat{R}(\theta) + \lambda \parallel \theta \parallel_1$$

Promotes **sparse** parameters.

Use cases:

- Feature selection
- Interpretable models
- High-dimensional problems (e.g., genomics)

3.3 Elastic Net

Combination of L1 and L2:

$$\hat{R}_{EN} = \hat{R}(\theta) + \lambda_1 \parallel \theta \parallel_1 + \lambda_2 \parallel \theta \parallel_2^2.$$

3.4 Early Stopping

Early stopping is equivalent to **implicit L2 regularization** in gradient descent.

Textual diagram:

Training Loss: \   (keeps decreasing)

Validation Loss: __/\ (minimum then rises)

               ^ stop here

---

4. Regularization in Neural Networks

4.1 Dropout

Randomly zeroing activations:

$$h_i' = \frac{m_i h_i}{1-p}, m_i \sim \text{Bernoulli}(1-p)$$

Interpretable as model averaging over 2^nsub-networks.

Effects:

- Reduces co-adaptation
- Introduces noise in training
- Strongly improves generalization

Subtleties:

- Should not be used in all layers of Transformers (e.g., too much dropout harms attention).

4.2 Batch Normalization as Implicit Regularizer

BN introduces **mini-batch noise**:

$$\hat{x} = \frac{x - \mu_{\text{batch}}}{\sigma_{\text{batch}}}$$

Noise comes from finite batch estimates of μ and σ.
This acts as a regularizer, particularly in CNNs.

4.3 Data Augmentation

Transforms $x \to \tilde{x} = T(x)$to enforce invariances.

Types:

- Geometric (flips, rotations)
- Color-space jitter
- Mixup / CutMix
- Text augmentations (back-translation, random spans)

Mixup formulation:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \tilde{y} = \lambda y_i + (1 - \lambda)y_j.$$

4.4 Label Smoothing

Softens one-hot labels:

$$y_{LS} = (1 - \alpha)y_{\text{onehot}} + \frac{\alpha}{K}$$

Reduces overconfidence and sharp minima.

---

5. Bayesian and Information-Theoretic Regularization

5.1 Maximum a Posteriori (MAP)

Regularized loss is equivalent to a MAP estimate:

- L2 ⇔ Gaussian prior
- L1 ⇔ Laplace prior

$$\theta_{MAP}^{*} = \arg\min_{\theta} \left(-\log p(D \mid \theta) - \log p(\theta)\right)$$

5.2 PAC-Bayesian Regularization

Bounds generalization error in terms of KL divergence between posterior and prior.

$$R(f) \leq \hat{R}(f) + \sqrt{\frac{KL(Q \parallel P) + \ln\left(2\sqrt{n}/\delta\right)}{2(n-1)}}$$

Used as theoretical basis for *Sharpness-Aware Minimization* (SAM) and flat-minima regularization.

5.3 Variational Regularization (e.g., VAEs)

KL term acts as regularizer:

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x \mid z)] - \beta\, KL(q(z \mid x) \parallel p(z))$$

---

6. Geometry-Based and Manifold Regularization

Assumes data lie on low-dimensional manifolds.
Regularizers enforce smoothness along the manifold.

Graph Laplacian regularization:

$$\Omega(f) = \sum_{i,j} w_{ij} \left(f(x_i) - f(x_j)\right)^2$$

---

7. Regularization in Modern Foundation Models

7.1 Transformers

Regularization stack includes:

- Dropout in:
    - attention weights
    - MLP layers
- Weight decay
- LayerNorm structure (implicit regularization)
- Stochastic depth (DeepNet, EfficientNet)

Attention dropout

During softmax attention:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Dropout applied before multiplying by V.

7.2 LLM/SLM Training Regularization

Modern LLMs use:

- **Worse-case-aware optimization** (SAM)
- **Gradient clipping**
- **Sequence-level dropout**
- **Tokenizer-level regularization**
- **Entmax or softened cross-entropy variants**

Instruction-tuned models additionally use:

- Preference optimization (DPO, ORPO) with implicit regularization
- KL penalty against base model

---

8. Comparison Across Methods

METHOD	ENCOURAGES	PROS	CONS	BEST USE CASES
L2	Smooth small weights	Stable, universal	Weak sparsity	All deep nets
L1	Sparsity	Feature selection	Hard to optimize	High-dimensional
DROPOUT	Robustness	Strong in CNNs/MLPs	Suboptimal in attention	Vision, tabular
BN	Smoother landscape	Helps optimization	Batch-size dependent	CNNs
DATA AUG.	Invariances	Big gains	Domain-specific	Vision, NLP
LABEL SMOOTHING	Low confidence	Prevents overfitting	Can hurt calibration	Classification
SAM	Flat minima	Strong generalization	Expensive	LLMs, vision
EARLY STOPPING	Implicit L2	Simple & cheap	Requires validation	All models

---

9. Implementation Insights

Choosing Weight Decay

- AdamW: weight decay ≈ 0.01 (standard for Transformers)
- SGD: ~1e-4 to 1e-2

Avoid Over-regularization

Symptoms: underfitting, vanishing gradients, overly smooth predictions.

Regularization Interactions

- Dropout + LayerNorm: careful tuning required.
- Weight decay + Adaptive optimizers: use *decoupled* variant (AdamW).
- Data augmentation + label smoothing: too much may degrade calibration.

---

10. Best-Practice Recommendations

For Vision

- Strong augmentations (RandAugment, Mixup, CutMix)
- Moderate dropout
- High weight decay for large nets
- Stochastic depth for deep architectures

For NLP

- Low dropout (0.1)
- Label smoothing (0.1)
- AdamW weight decay (0.01)
- Gradient clipping
- Data diversification (span masking, token perturbation)

For LLM Fine-Tuning

- KL regularization against base model
- Low learning rates
- Avoid excessive dropout — harms previously learned representations
- Consider LoRA rank constraints as structural regularization

---

11. Case Study

A. Image Classification (ResNet-50 on CIFAR-100)

Setup

- SGD + momentum
- Weight decay = 5e-4
- Data augmentation = horizontal flips & random crops
- Label smoothing = 0.1
- Mixup = α = 0.2
- Stochastic depth = 0.1

Observed Outcomes

- +6–8% top-1 accuracy improvement
- Training loss decreases more slowly, but validation accuracy increases steadily
- Learned features smoother, more robust to adversarial noise

---

B. LLM Fine-Tuning (7B Model for Instruction Following)

Regularization Stack

- KL penalty: 0.1
- Weight decay: 0.01
- Low dropout: 0.0–0.1
- Gradient norm clipping: 1.0
- Sequence packing for efficiency (not a regularizer but interacts with loss dynamics)

Outcomes

- Prevents catastrophic forgetting
- Reduces overfitting to specific instruction styles
- Maintains fluency and factuality better than unregularized fine-tuning

---

Conclusion

Regularization is fundamental to building robust machine learning systems.
The modern landscape integrates **explicit**, **implicit**, **structural**, and **Bayesian** regularization methods.
A rigorous approach—combining theoretical understanding with empirical best practices—enables practitioners to train models that generalize reliably across domains.