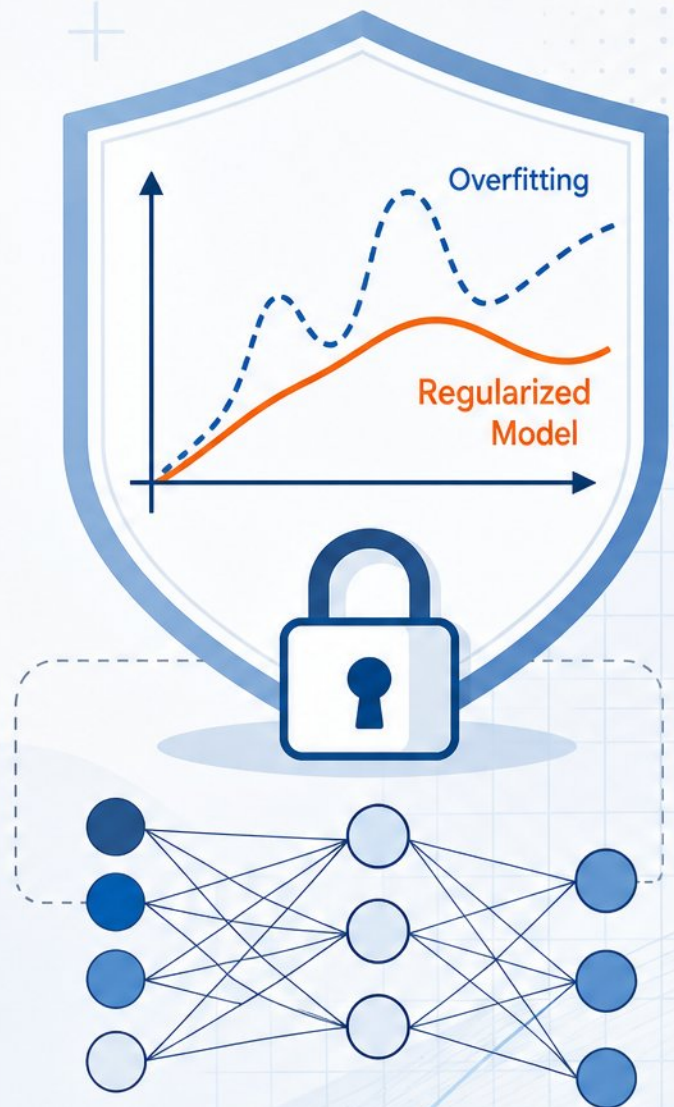


Regularization in Machine Learning: Concepts, Purpose, and Examples

- Family of techniques reducing overfitting via constraints or penalties
- Trades small training error increase for better generalization on unseen data
- Techniques covered: L1, L2, Elastic Net, Dropout, Early Stopping
- Goal: explain why regularization works and choose a reasonable technique

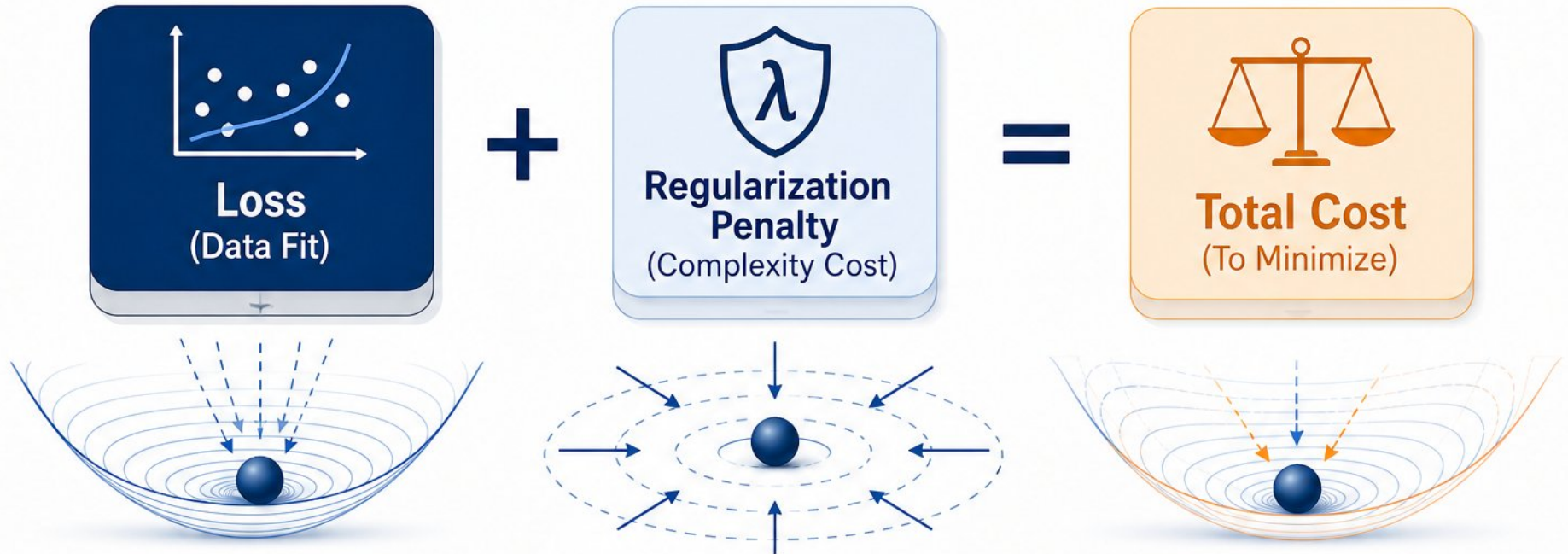


Why Regularization Is Needed

- Bias-variance tradeoff: simple assumptions vs. sensitivity to training data
- Complex models can memorize noise, then fail on unseen data
- Example: high-degree polynomial fitting a small dataset
- Warning sign: low training error, large validation/test error gap



The Core Idea: Adding a Penalty Term



-  • Total cost = loss + regularization penalty
-  • Penalty grows as model complexity increases
-  • Examples: sum of squared or absolute weights
-  • Model balances data fit against small weights
-  • Regularization favors simpler, more stable solutions



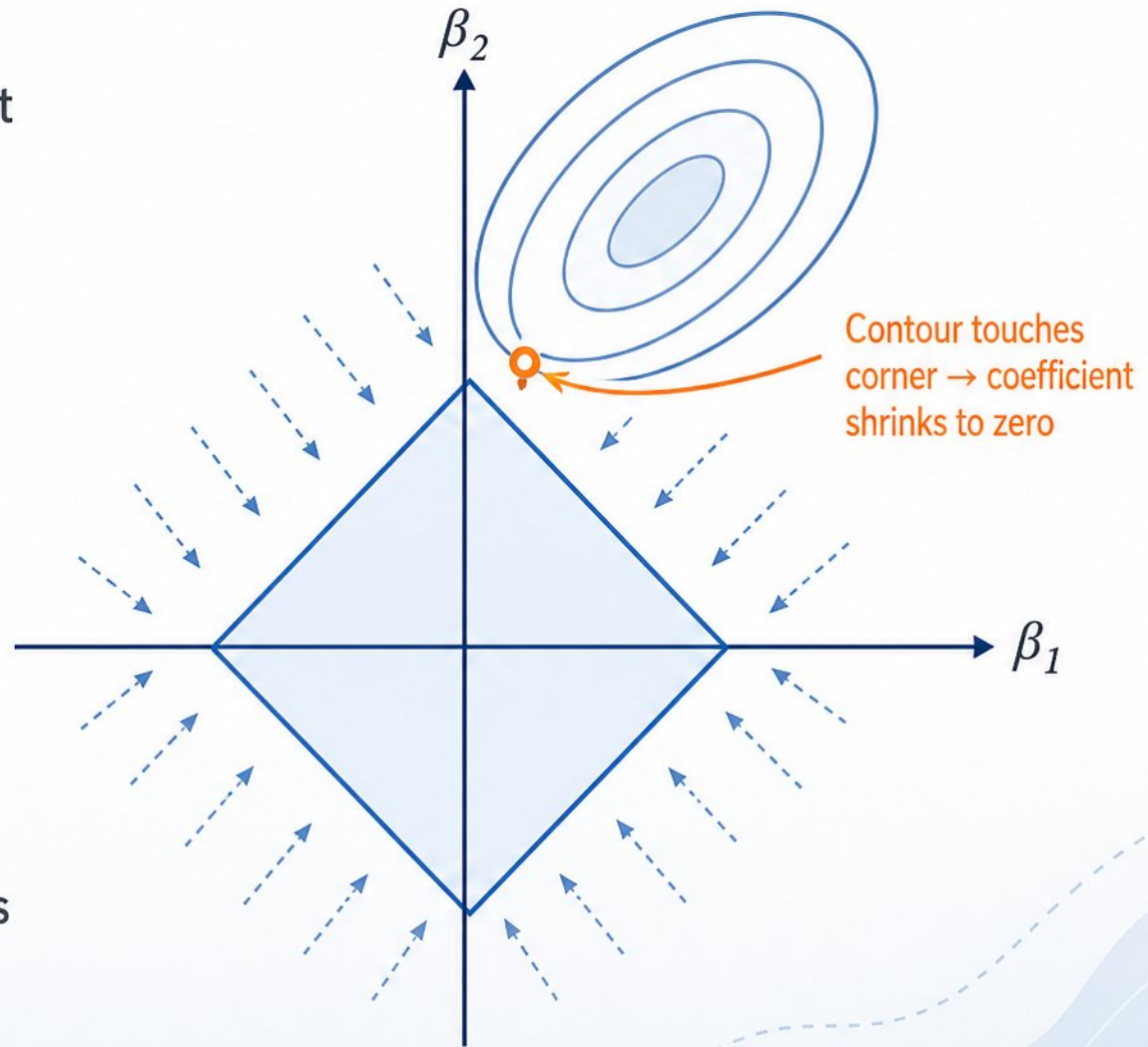
The Regularization Parameter: Why It Matters

- Lambda controls regularization strength; scikit-learn often calls it alpha
- Small lambda: weak penalty, model may overfit training data
- Large lambda: strong penalty, model may underfit and miss patterns
- The goal is balancing training fit against generalization
- Cross-validation is the standard method for choosing a good lambda



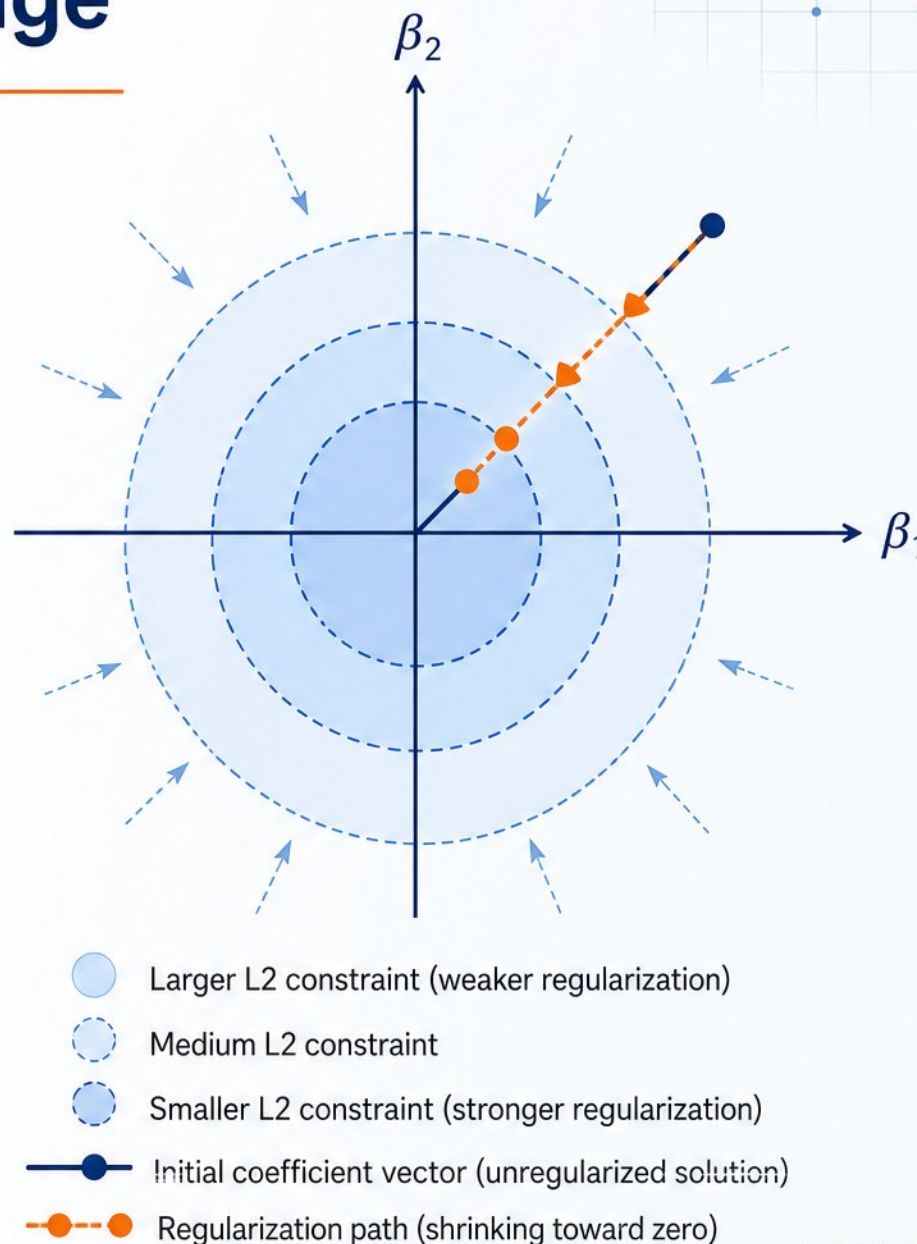
L1 Regularization: Lasso

- ▶ Adds sum of absolute coefficient values to the loss
- ▶ Can shrink some coefficients exactly to zero
- ▶ Produces sparse models with automatic feature selection
- ▶ Best for high-dimensional data or mostly irrelevant features
- ▶ Diamond-shaped constraint region encourages corner solutions



L2 Regularization: Ridge

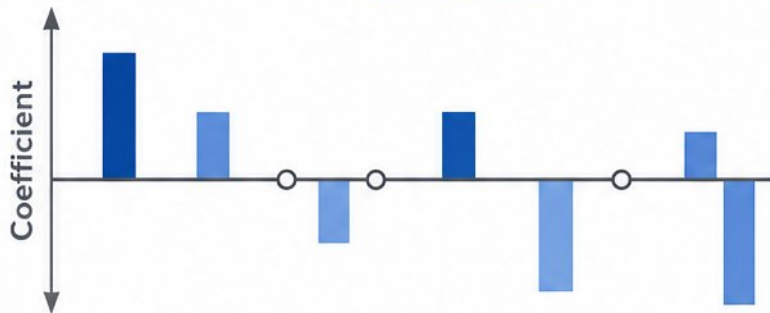
- Adds squared coefficient values to the loss
- Shrinks coefficients toward zero, never exactly zero
- Handles multicollinearity by distributing weight across correlated features
- In neural networks, the same idea is called weight decay
- Example: Ridge regression for stable, dense models



Comparing L1 and L2

L1 (Lasso)

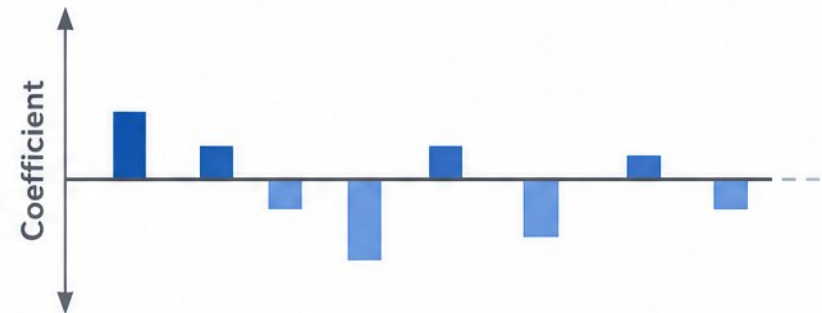
Forces some coefficients to exactly zero



Exact zeros → Sparse model

L2 (Ridge)

Keeps all features, shrinks weights toward zero



No exact zeros → Dense model



- L1 (Lasso): forces some coefficients to exactly zero



- L2 (Ridge): keeps all features, shrinks weights toward zero



- Prefer Lasso for irrelevant features or interpretability



- Prefer Ridge when most features contribute or are correlated



- Lasso can arbitrarily drop correlated features — Elastic Net fixes this

Elastic Net: Combining L1 and L2



- Weighted combination of L1 and L2 penalties



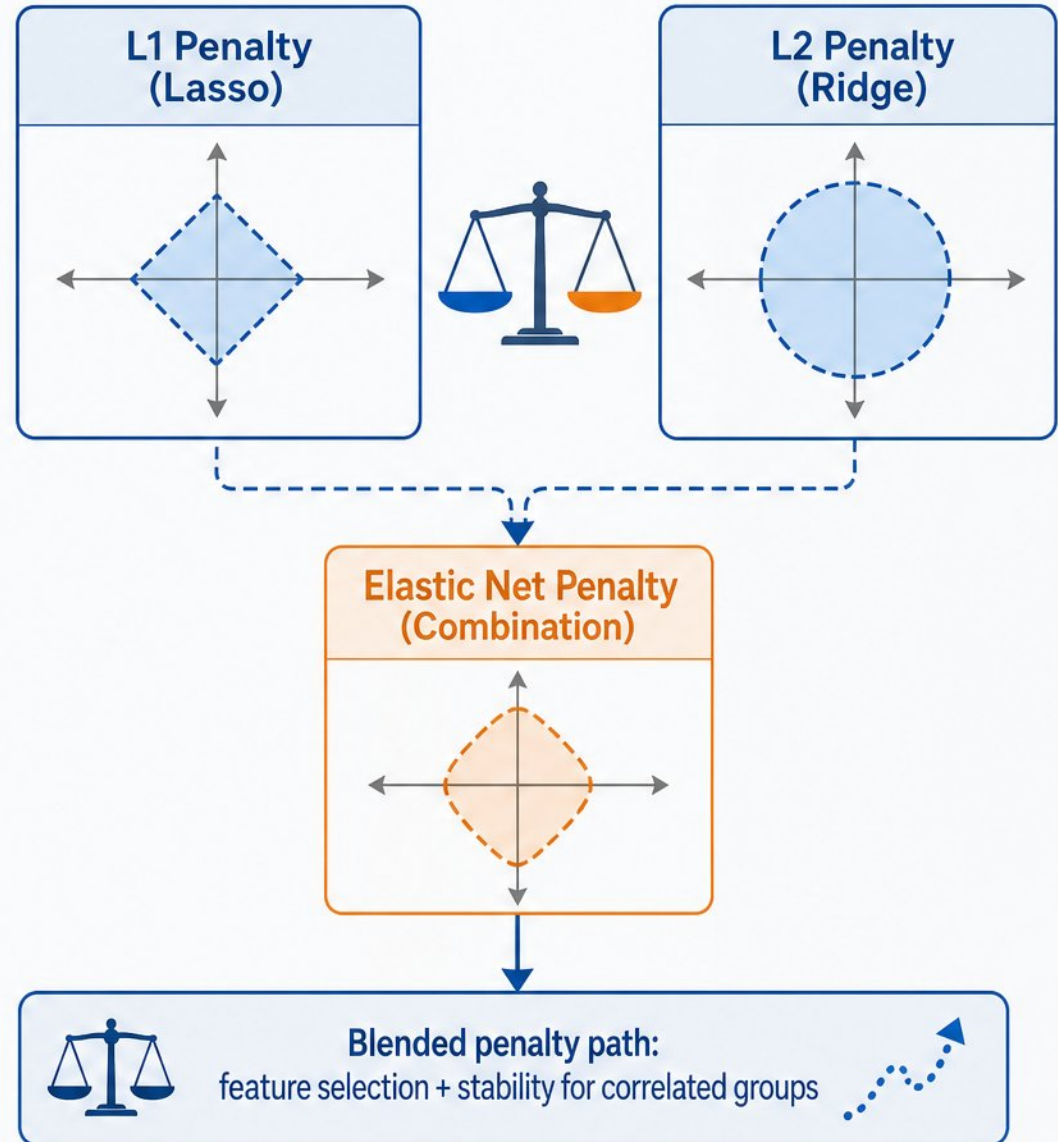
- Alpha controls overall strength; l1_ratio controls the mix



- Preserves Lasso's feature selection while stabilizing correlated groups



- Often a good default when p exceeds n or many features are correlated



Regularization Beyond Linear Models



- **Weight decay:** L2 regularization applied to network weights



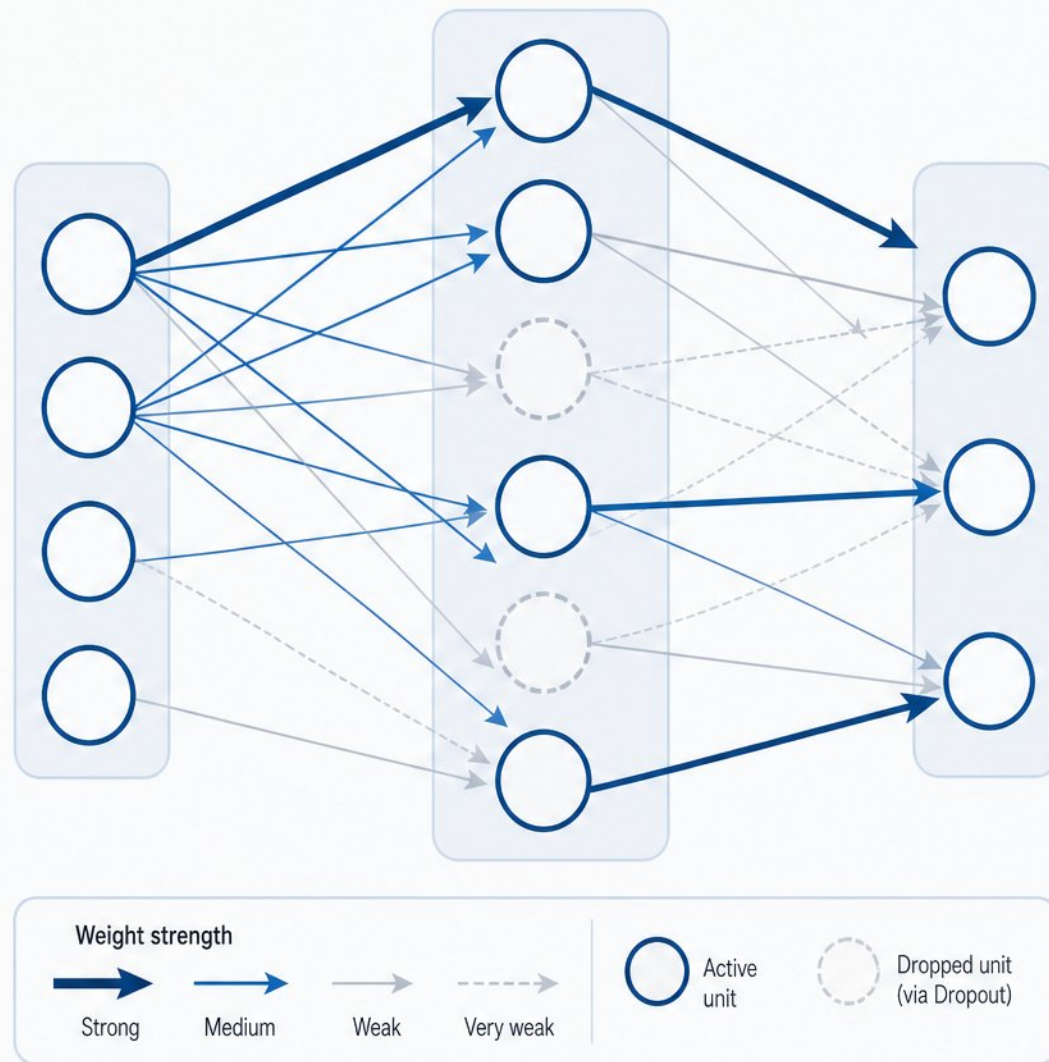
- **Dropout:** randomly drops units during training to reduce neuron reliance



- Dropout helps small datasets and complex networks most

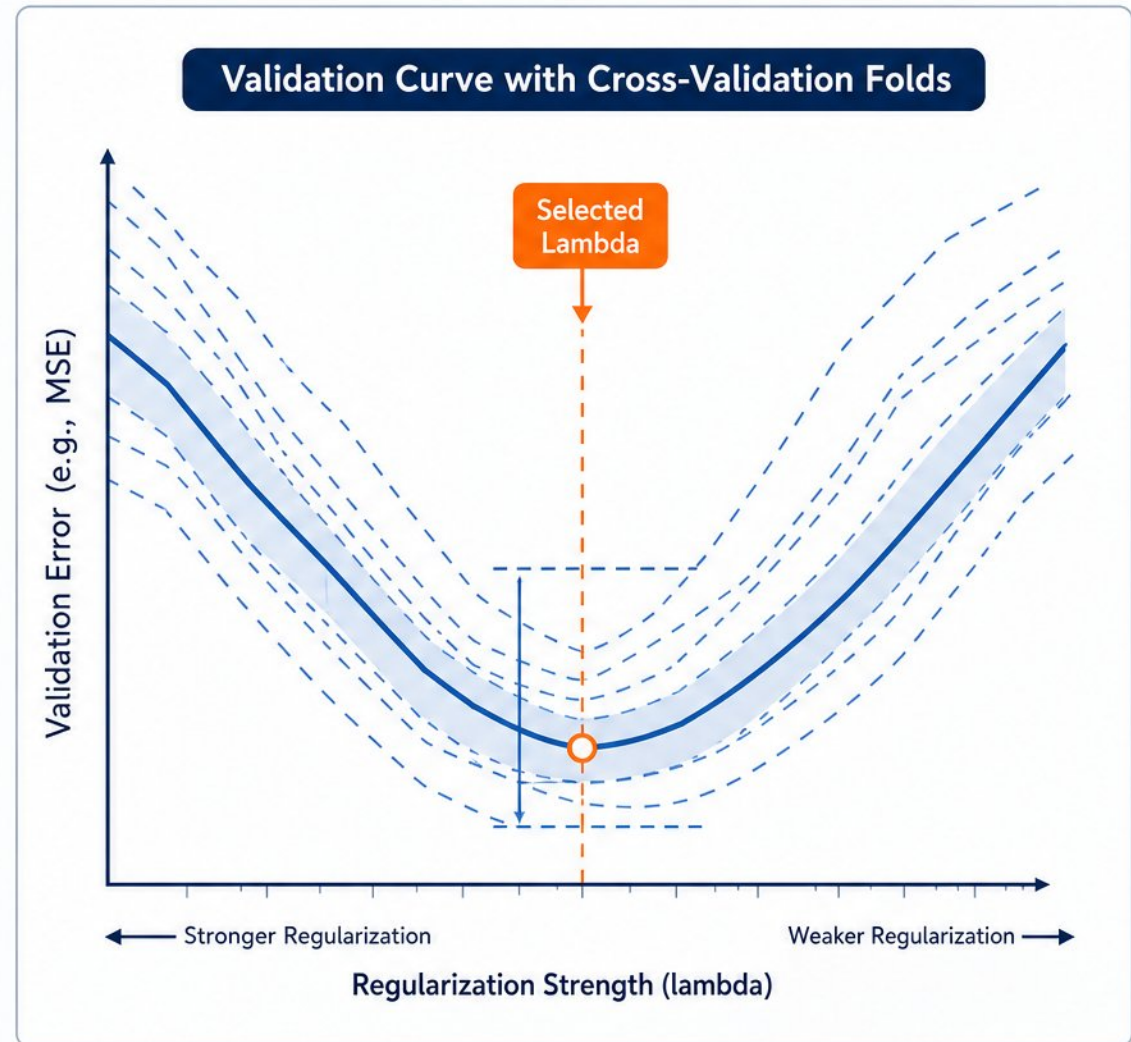


- **Early stopping:** halt training when validation performance worsens



Practical Strategy: Tuning and Scaling

- ✓ Select lambda through cross-validation as part of model selection
- ✓ Standardize features before applying regularized linear models
- ✓ Never tune hyperparameters on the held-out test set
- ✓ Use validation set or nested cross-validation instead
- ✓ One standard error rule: choose simpler model within 1 SE



Common Pitfalls and How to Avoid Them



- Over-regularization causes underfitting; under-regularization allows overfitting



- Do not regularize a model that is already underfitting



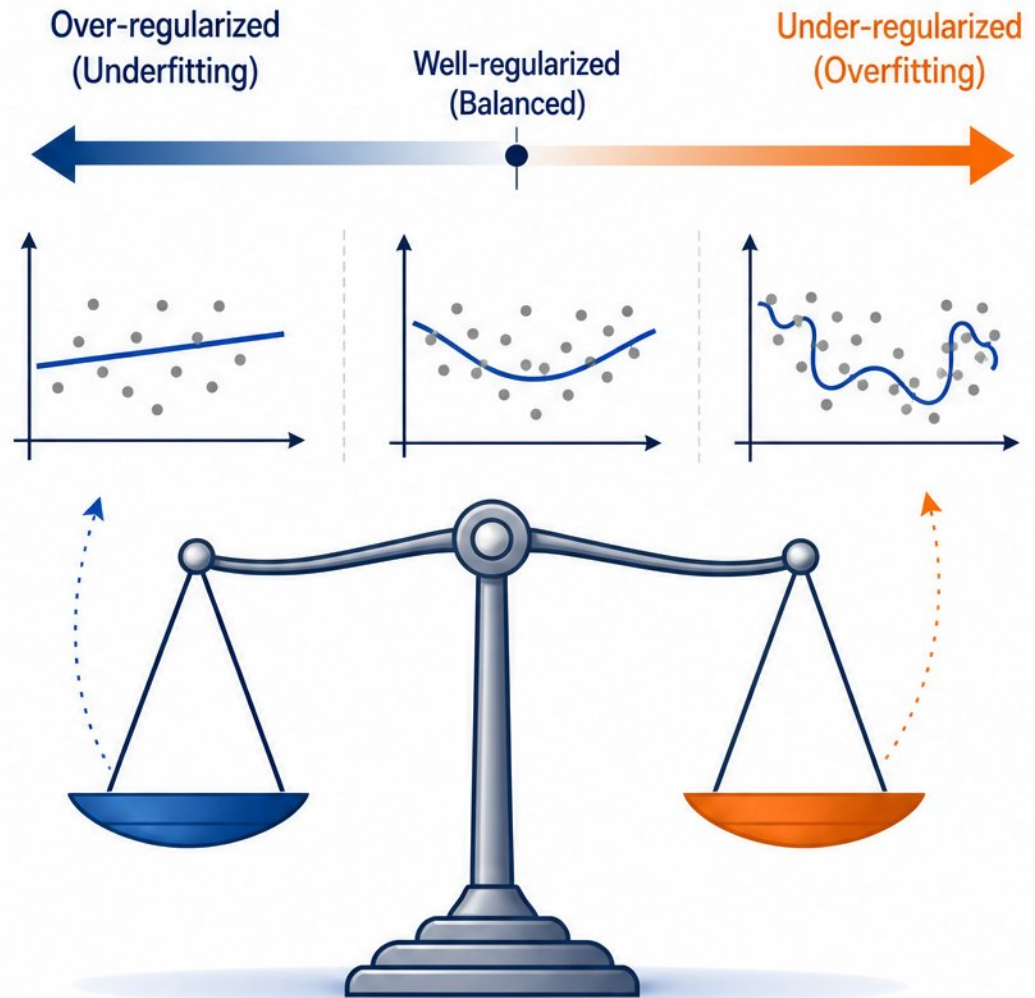
- Lasso instability: different features selected across folds or seeds



- Compare unregularized and regularized performance; use validation curves



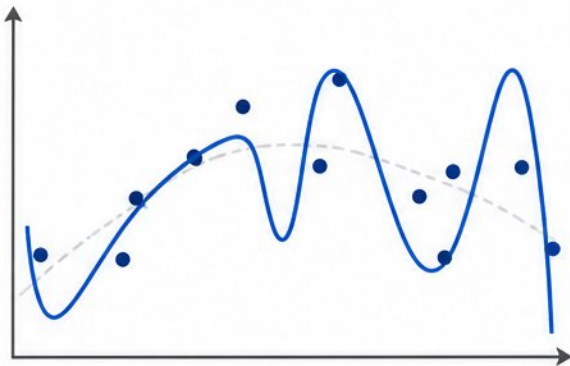
- Never evaluate only training performance



Worked Example: Linear Regression



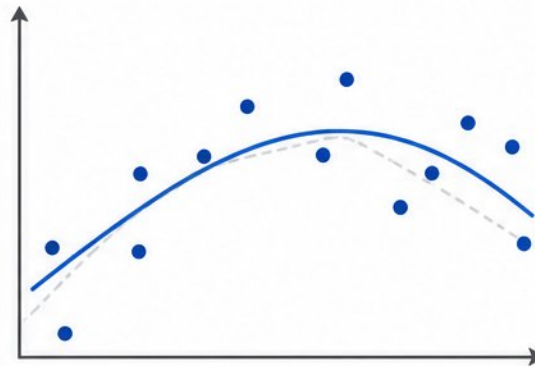
OLS (Unregularized)



Very flexible fit that follows noise in the data.



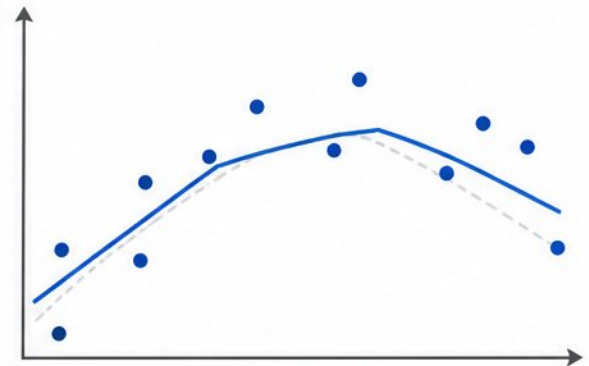
Ridge Regression



Shrinks coefficients; smoother curve with less variance.



Lasso Regression



Many coefficients set to zero; simpler, more interpretable fit.

- ✓ Overfitting appears when many polynomial features are added to OLS
- ✓ OLS coefficients grow large; Ridge shrinks them toward zero
- ✓ Lasso forces most coefficients exactly to zero
- ✓ Compare train vs. test error: regularization closes the generalization gap



Worked Example: Regularizing a Neural Network



- Small noisy dataset causes an overfit network



- Dropout randomly disables units during training



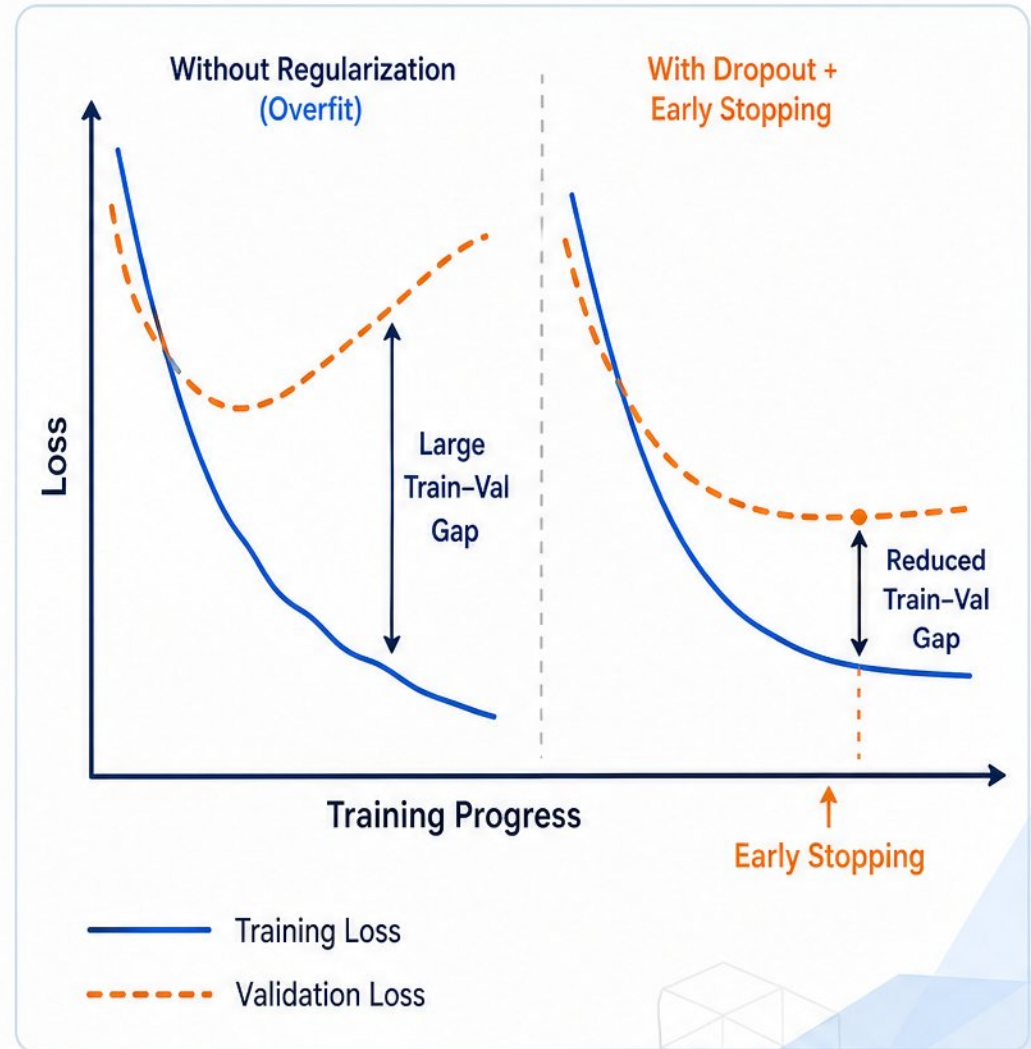
- Training and validation curves show a reduced gap



- Early stopping halts training when validation loss worsens



- Dropout changes the structure; weight decay shrinks weights



Summary and Next Steps



Regularization trades a small bias increase for much lower variance.



Key techniques: Lasso, Ridge, Elastic Net, Dropout, Early Stopping.



Choose based on feature relevance, correlation, and dataset size.

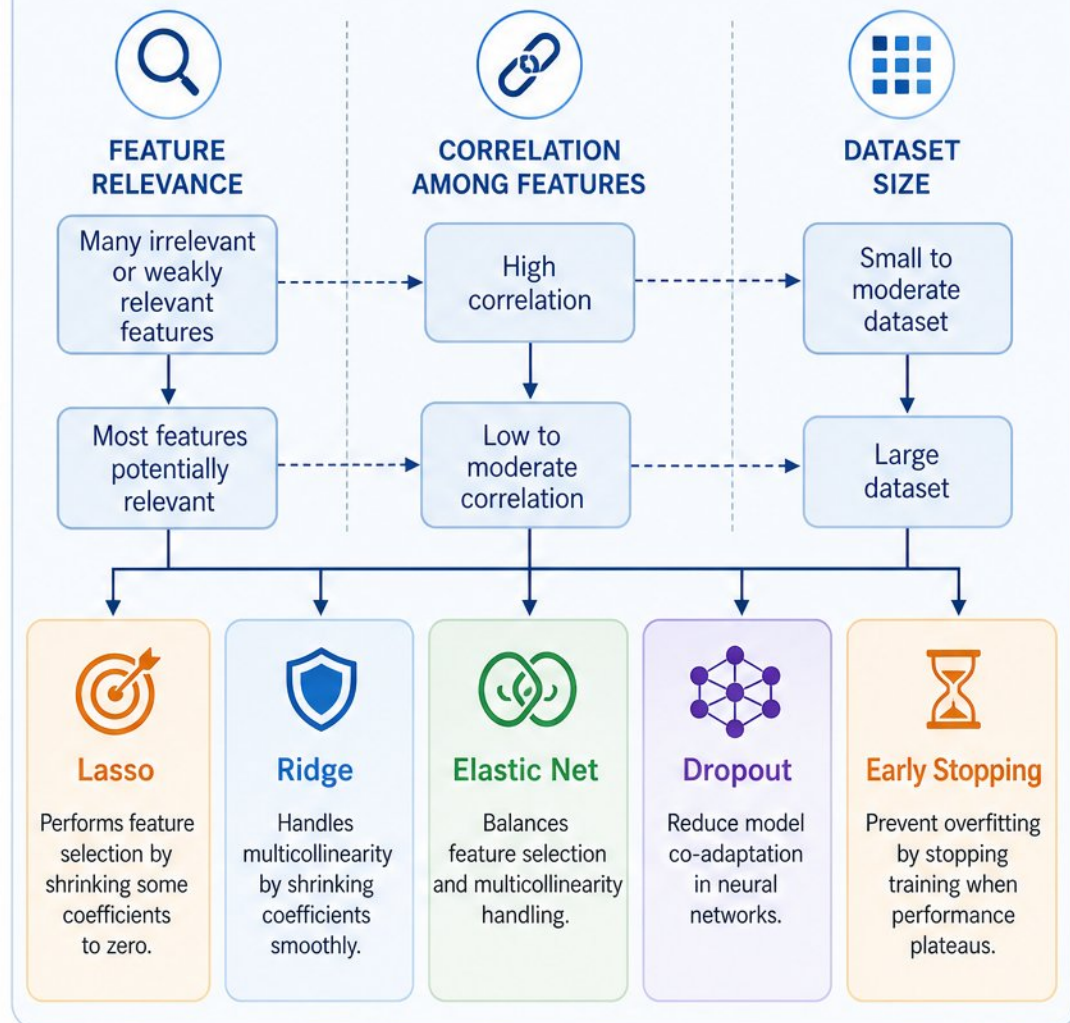


Next: tune strength with cross-validation and compare train/test error.



Practice using beginner-friendly scikit-learn and Keras tutorials.

DECISION GUIDE: MAP DATA CHARACTERISTICS TO REGULARIZATION CHOICES



PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/64ff34b6/public