Model Selection and Evaluation

CMSC 173 - Machine Learning

Course Lecture

Outline

Introduction to Model Selection

The Model Selection Problem

Central Question: How do we choose the best model?

Challenges

- Multiple algorithms available
- Different hyperparameters
- Trade-offs between complexity and performance
- Avoiding overfitting
- Generalization to unseen data

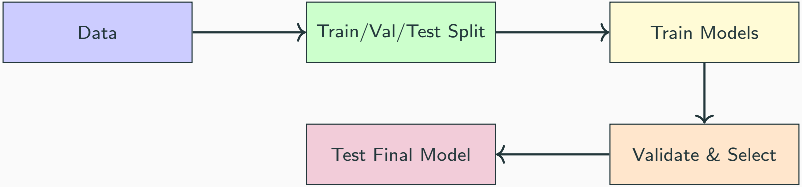
Goals

- Select optimal model architecture
- Tune hyperparameters effectively
- Ensure reliable performance
- Balance bias and variance
- Maximize generalization

Key Insight

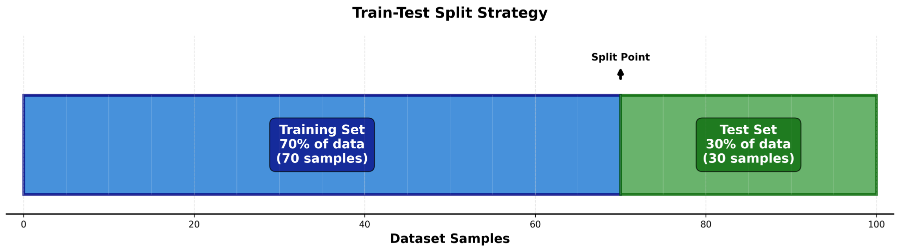
Model selection is not just about training performance, but about how well the model generalizes to new, unseen data.

Model Selection Pipeline



- **Split:** Divide data into training, validation, and test sets
- **Train:** Fit multiple candidate models
- **Validate:** Compare models on validation set
- **Select:** Choose best performing model
- **Test:** Final evaluation on held-out test set

Train-Validation-Test Split



Training Set

- Model fitting
- Learning parameters
- 60-70% of data

Validation Set

- Model selection
- Hyperparameter tuning
- 15-20% of data

Test Set

- Final evaluation
- Unbiased estimate
- 15-20% of data

Bias-Variance Decomposition

Understanding Prediction Error

For a regression problem, the expected prediction error can be decomposed:

Error Decomposition

$$\mathbb{E}[(y - \hat{f}(x))^2] = \text{Bias}^2[\hat{f}(x)] + \text{Var}[\hat{f}(x)] + \sigma^2$$

Bias

$$\text{Bias}[\hat{f}] = \mathbb{E}[\hat{f}] - f$$

Error from wrong assumptions in the learning algorithm

Variance

$$\text{Var}[\hat{f}] = \mathbb{E}[(\hat{f} - \mathbb{E}[\hat{f}])^2]$$

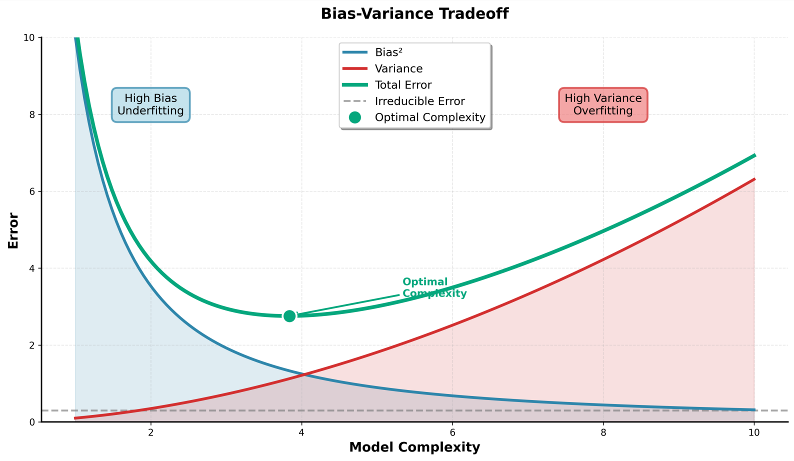
Error from sensitivity to training set variations

Irreducible Error

$$\sigma^2 = \text{Var}[\epsilon]$$

Noise in the data that cannot be reduced

The Bias-Variance Tradeoff



Key Insight

- As model complexity increases, bias decreases but variance increases
- The optimal model minimizes the total error (bias^2 + variance)
- There exists a sweet spot that balances both sources of error

High Bias vs High Variance

High Bias (Underfitting)

Characteristics:

- Overly simple model
- Poor training performance
- Poor test performance
- Cannot capture data patterns

Solutions:

- Increase model complexity
- Add more features
- Reduce regularization
- Train longer

High Variance (Overfitting)

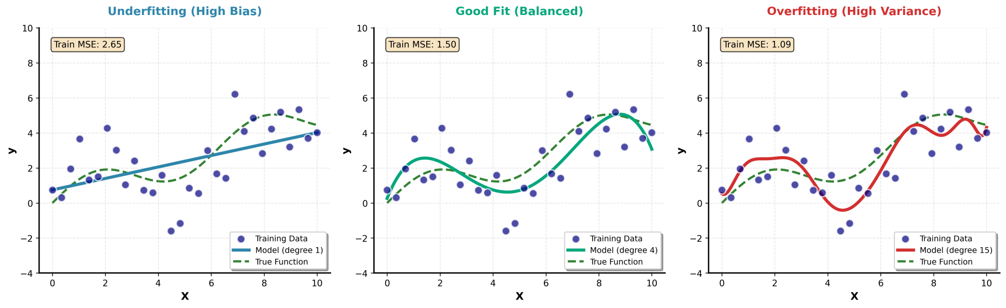
Characteristics:

- Overly complex model
- Excellent training performance
- Poor test performance
- Memorizes training data

Solutions:

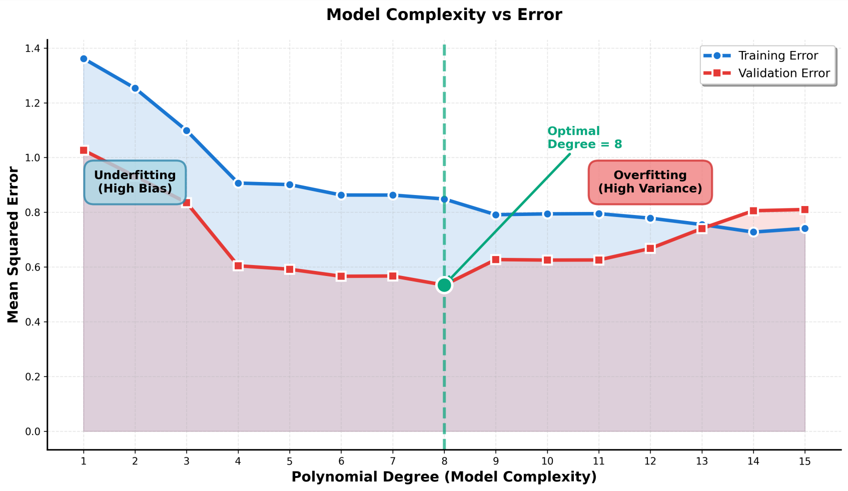
- Simplify model
- Get more training data
- Increase regularization
- Use early stopping

Visualizing Underfitting and Overfitting



- **Left:** Underfitting - linear model cannot capture nonlinear relationship
- **Center:** Good fit - balanced complexity captures true pattern
- **Right:** Overfitting - high-degree polynomial fits noise

Model Complexity and Error



Observations

- Training error decreases monotonically with complexity
- Validation error has a U-shaped curve
- Gap between curves indicates overfitting
- Optimal complexity minimizes validation error

Model Validation and Evaluation

Why Do We Need Validation?

The Fundamental Problem

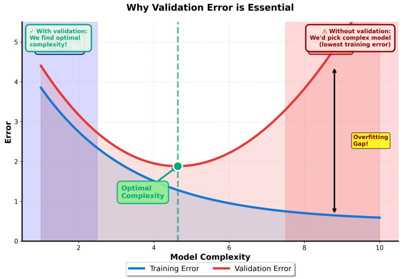
We cannot evaluate model performance on the same data used for training!

Training Error is Optimistic

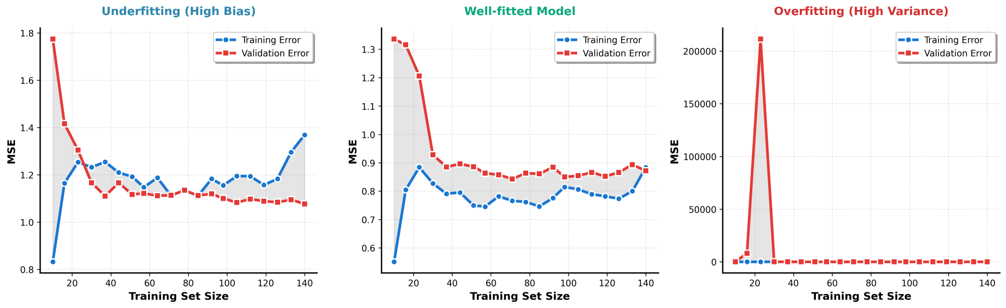
- Model has seen the training data
- Can memorize patterns and noise
- Does not reflect generalization
- Always decreases with complexity

Validation Error is Realistic

- Model has not seen validation data
- Measures true generalization
- Enables fair model comparison
- Guides hyperparameter selection



Learning Curves



- **Underfitting:** Both errors high, converge to high value
- **Well-fitted:** Both errors low, small gap between them
- **Overfitting:** Large gap between training and validation error

Cross-Validation: Motivation

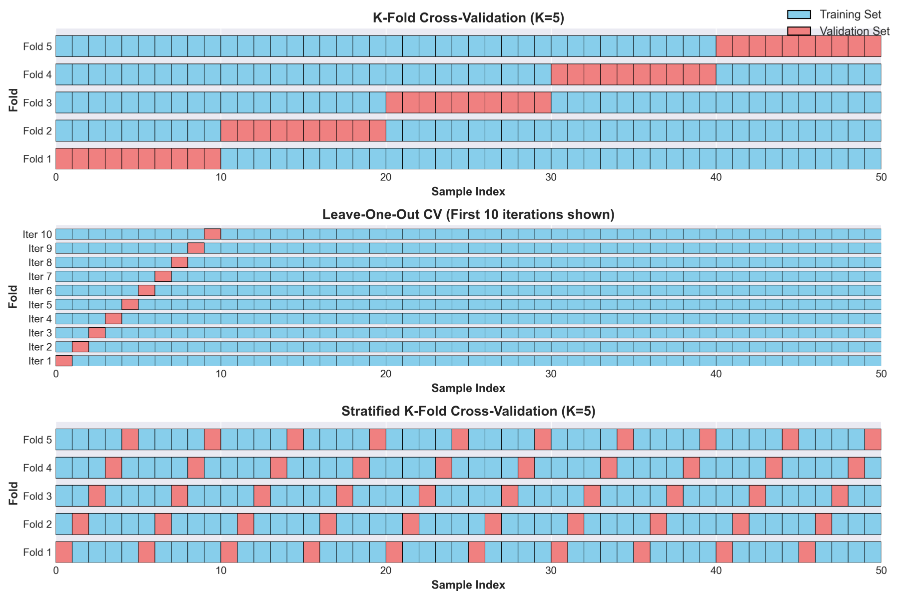
Problem with Single Train-Val Split

- Results depend on random split
- Some data points never used for training
- Some never used for validation
- High variance in performance estimates

Cross-Validation Solution

- Use multiple train-validation splits
- Every data point used for both training and validation
- Average results across splits for robust estimate
- Reduces variance in performance evaluation

Cross-Validation Schemes



K-Fold CV

Stratified K-Fold

K-Fold Cross-Validation Algorithm

Algorithm 1 K-Fold Cross-Validation

Require: Dataset D, Model M, Number of folds K
Ensure: Cross-validation score

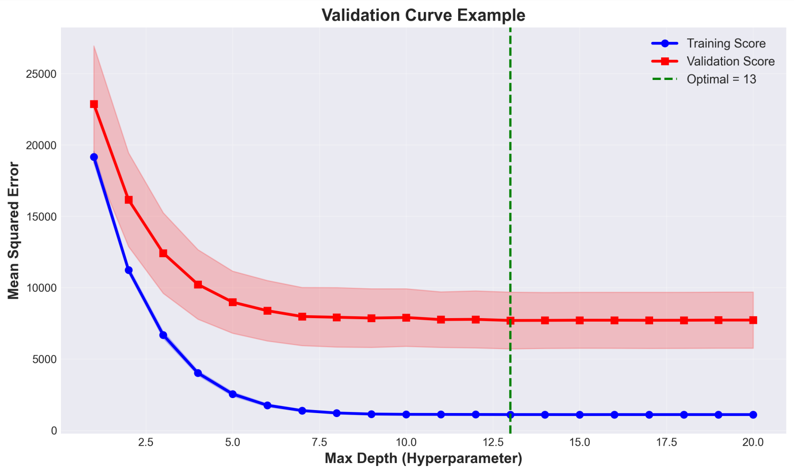
```
Randomly partition D into K equal-sized subsets D_1, D_2, ..., D_K
Initialize scores = []
for i = 1 to K do
    D_val ← D_i
    D_train ← D \ D_i
    Train model M̂ on D_train
    s_i ← Evaluate(M̂, D_val)
    Append s_i to scores
end for
return (1/K) Σ_{i=1}^{K} s_i
```

$$\frac{1}{K}\sum_{i=1}^{K} s_i$$

Common Choices

$K = 5$ or $K = 10$ are typical values balancing computational cost and variance reduction.

Validation Curve

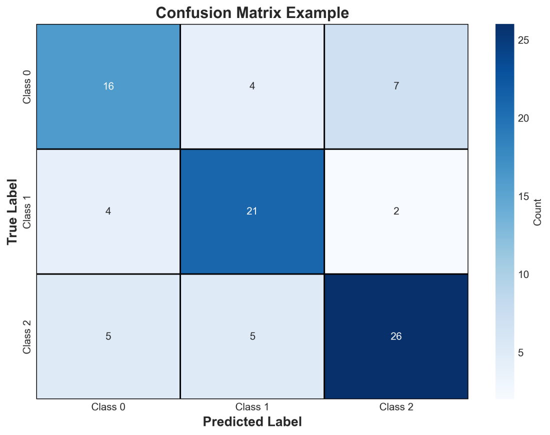


Using Validation Curves

- Plot training and validation scores vs. hyperparameter values
- Identify optimal hyperparameter setting
- Diagnose underfitting and overfitting regions
- Select model with best validation performance

Evaluation Metrics

Classification Metrics: Confusion Matrix



Definitions

- **TP:** True Positives
- **TN:** True Negatives
- **FP:** False Positives (Type I error)
- **FN:** False Negatives (Type II error)

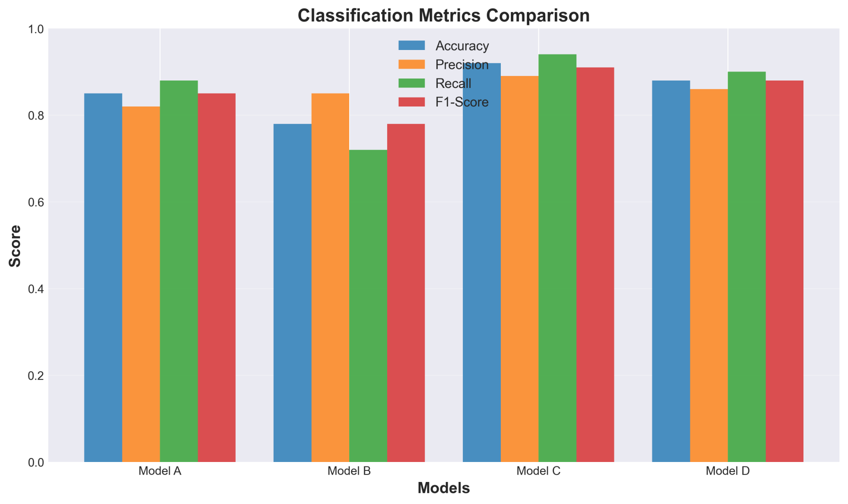
Key Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Classification Metrics Comparison



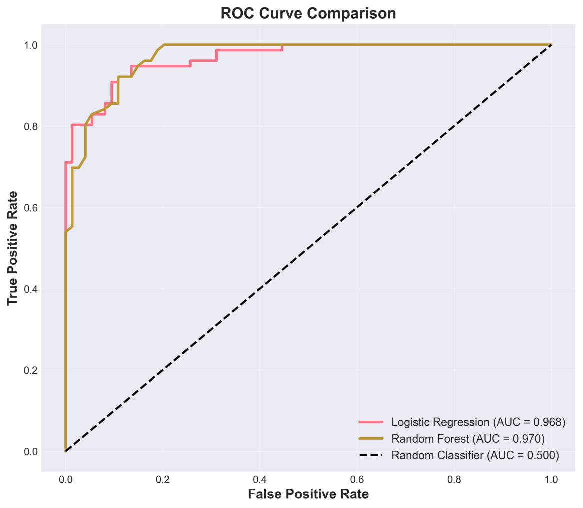
Accuracy

Overall correctness; can be misleading with imbalanced classes

F1-Score

Harmonic mean of precision and recall:
$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

ROC Curve and AUC



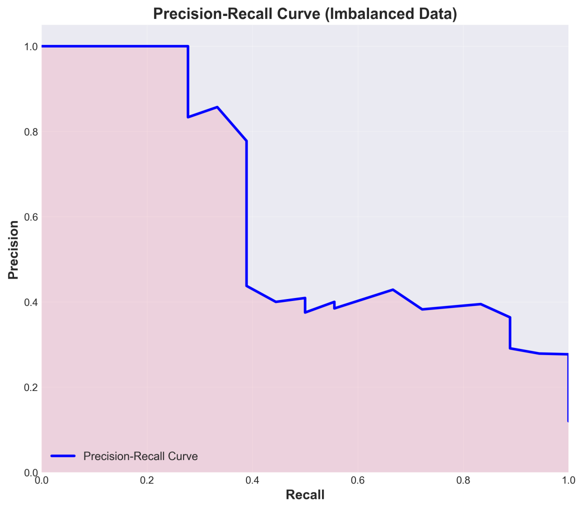
ROC Curve

- Plots TPR vs FPR
- Shows performance across thresholds
- Diagonal = random classifier
- Upper-left corner = perfect

AUC Score

- Area Under ROC Curve
- Range: [0, 1]
- 0.5 = random
- 1.0 = perfect
- Threshold-independent

Precision-Recall Curve



When to Use

- Imbalanced datasets
- Care about positive class
- False positives costly
- Alternative to ROC

Interpretation

- High area = good performance
- Trade-off between precision and recall
- Choose threshold based on application needs

Regression Metrics

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Penalizes large errors heavily
- Same units as y^2
- Always non-negative
- Lower is better

Root Mean Squared Error

$$\text{RMSE} = \sqrt{\text{MSE}}$$

- Same units as y
- More interpretable than MSE

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Robust to outliers
- Same units as y
- Easy to interpret

R-Squared (R^2)

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

- Proportion of variance explained
- Range: $(-\infty, 1]$
- $1 =$ perfect predictions

Regularization

What is Regularization?

Definition

Regularization is a technique to prevent overfitting by adding a penalty term to the loss function that discourages complex models.

General Form

$$\text{Loss}_{\text{regularized}} = \text{Loss}_{\text{data}} + \lambda \cdot \text{Penalty}(\text{parameters})$$

where $\lambda \geq 0$ is the **regularization parameter** controlling the strength of regularization.

Benefits

- Reduces overfitting
- Improves generalization
- Encourages simpler models
- Can perform feature selection

Trade-off

- λ too small: underfitting
- λ too large: underfitting
- Must tune λ via validation

Ridge Regression (L2 Regularization)

Objective Function

$$\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2$$

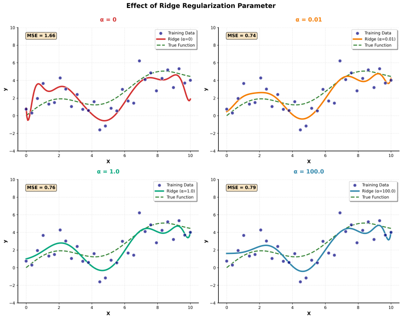
where $\|\mathbf{w}\|_2^2 = \sum_{j=1}^{d} w_j^2$ is the L2 norm.

Characteristics

- Shrinks coefficients towards zero
- Does not set coefficients exactly to zero
- Has closed-form solution
- Stable and computationally efficient
- Preferred when all features are relevant

Solution

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$



Lasso Regression (L1 Regularization)

Objective Function

$$\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_1$$

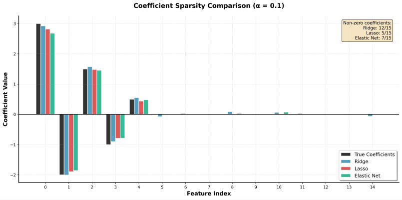
where $\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$ is the L1 norm.

Characteristics

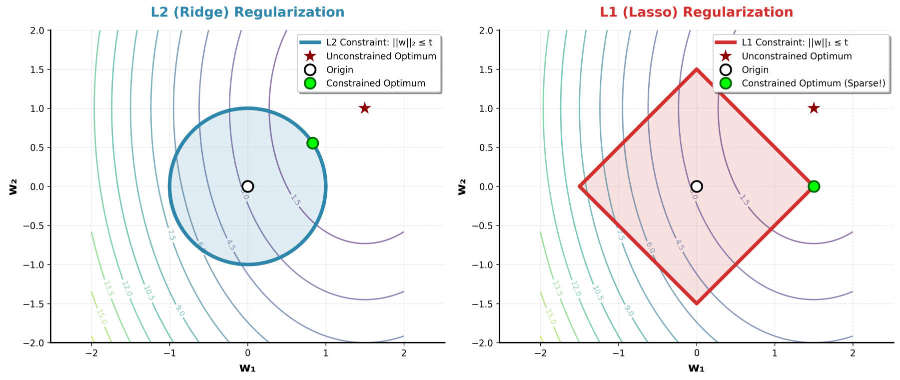
- Can set coefficients exactly to zero
- Performs automatic feature selection
- Produces sparse models
- No closed-form solution (use optimization)
- Preferred with many irrelevant features

Sparsity Property

Lasso's ability to zero out coefficients makes it ideal for interpretable models and high-dimensional data.

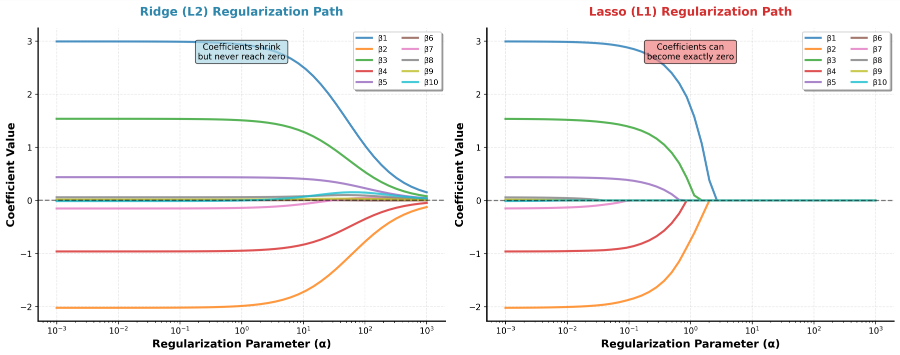


L1 vs L2 Regularization: Geometric Interpretation



- **L2 (Ridge):** Circular constraint region - solution rarely at axes (non-sparse)
- **L1 (Lasso):** Diamond constraint region - corners encourage sparse solutions
- Contours represent loss function, constraint region represents penalty

Regularization Paths



Observations

- **Ridge:** Coefficients shrink smoothly but never reach exactly zero
- **Lasso:** Coefficients can become exactly zero at finite λ
- As $\lambda \to \infty$, all coefficients approach zero
- Different coefficients zero out at different λ values in Lasso

Elastic Net: Combining L1 and L2

Objective Function

$$\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2$$

Alternatively parameterized with mixing parameter $\alpha \in [0, 1]$:

$$\text{Penalty} = \lambda \left[\alpha \|\mathbf{w}\|_1 + (1 - \alpha) \|\mathbf{w}\|_2^2\right]$$

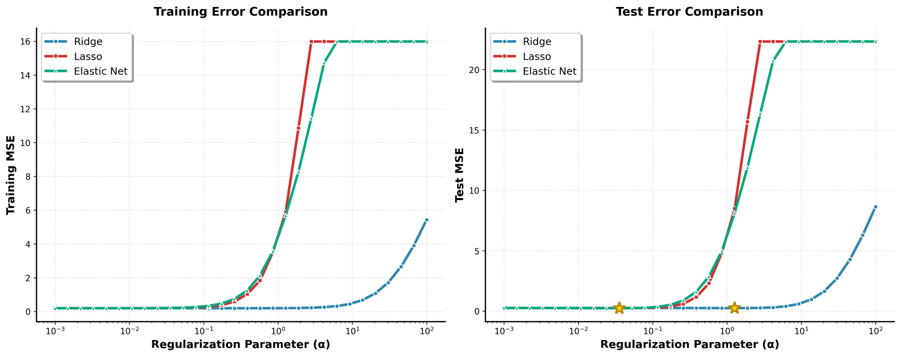
Advantages

- Combines benefits of Ridge and Lasso
- Handles correlated features better than Lasso
- Can select groups of correlated features
- More stable than Lasso

When to Use

- Many correlated features
- Want feature selection and grouping
- Lasso is too aggressive
- Ridge is not sparse enough

Comparing Regularization Methods



- All methods converge to similar training error with strong regularization
- Test error differences reveal generalization capabilities
- Optimal λ differs across methods

Regularization in Other Models

Neural Networks

- **Weight decay:** L2 penalty on weights
- **Dropout:** Randomly drop neurons during training
- **Early stopping:** Stop training before overfitting
- **Batch normalization:** Normalize activations

Support Vector Machines

- C parameter controls regularization
- Small C = strong regularization
- Large C = weak regularization

Decision Trees/Forests

- Max depth
- Min samples per leaf
- Max number of features
- Pruning

General Strategies

- Data augmentation
- Feature selection
- Ensemble methods
- Cross-validation for tuning

Best Practices and Common Pitfalls

Model Selection Best Practices

Do's

- Always use separate train/validation/test sets
- Use cross-validation for robust estimates
- Tune hyperparameters only on validation data
- Report final performance on test set (once!)
- Standardize/normalize features appropriately
- Use stratified splits for classification
- Track both training and validation metrics
- Document all preprocessing steps

Common Pitfalls to Avoid

Don'ts

- **Data leakage:** Including test data in preprocessing
- **Peeking at test set:** Multiple evaluations on test set
- **Ignoring class imbalance:** Using accuracy on imbalanced data
- **Not checking assumptions:** Assuming i.i.d. data
- **Overfitting validation set:** Excessive hyperparameter tuning
- **Cherry-picking results:** Reporting only best-case performance
- **Inadequate splitting:** Too small validation/test sets
- **Comparing on training data:** Always compare on validation

Data Leakage: A Critical Issue

What is Data Leakage?

Information from the test/validation set leaking into the training process, leading to overly optimistic performance estimates.

Common Sources

- Normalization using all data
- Feature selection on all data
- Imputation using all data
- Temporal data ordering issues
- Duplicate samples across splits

Prevention

- Split data FIRST
- Fit preprocessing only on training
- Transform validation/test separately
- Use pipelines
- Be careful with time series

Example: Correct Order

1. Split data → 2. Fit scaler on train → 3. Transform train/val/test → 4. Train model

Hyperparameter Tuning Strategies

Grid Search

- Exhaustive search over grid
- Guarantees finding best in grid
- Exponential in # parameters
- Good for few parameters

Random Search

- Randomly sample combinations
- Often finds good solutions faster
- Better for many parameters
- Can set computational budget

Bayesian Optimization

- Models objective function
- Guides search intelligently
- Most sample-efficient
- Good for expensive models

Practical Tips

- Start with coarse grid
- Refine around best values
- Use log scale for λ
- Parallelize when possible

Nested Cross-Validation

Problem

Using CV for both model selection and performance estimation gives biased results!

Solution: Nested CV

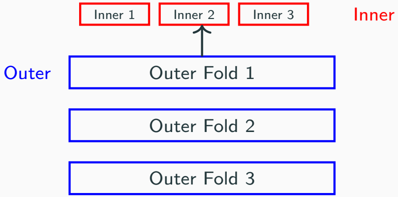
- **Outer loop:** Estimates true performance
- **Inner loop:** Selects hyperparameters
- Provides unbiased performance estimate
- More computationally expensive

Structure

For each outer fold:

1. Set aside test fold
2. Use inner CV to select hyperparameters
3. Train final model with best hyperparameters
4. Evaluate on test fold

Average outer fold results



Model Selection Checklist

Before Training

- ☐ Understand the problem and data
- ☐ Check for class imbalance
- ☐ Handle missing values
- ☐ Split data properly
- ☐ Standardize/normalize features
- ☐ Choose appropriate metrics

During Training

- ☐ Use cross-validation
- ☐ Track train and validation metrics
- ☐ Try multiple model types
- ☐ Tune hyperparameters systematically
- ☐ Check for overfitting

After Training

- ☐ Evaluate on test set (once!)
- ☐ Compare multiple metrics
- ☐ Analyze errors/confusion matrix
- ☐ Check for biases
- ☐ Document results
- ☐ Assess computational requirements

Golden Rule

Never touch the test set until final evaluation, and evaluate on it only **once**!

Summary and Key Takeaways

Summary: Key Concepts

1. **Bias-Variance Tradeoff**
    - Balance between model complexity and generalization
    - Underfitting (high bias) vs Overfitting (high variance)
2. **Model Validation**
    - Always use separate train/validation/test sets
    - Cross-validation provides robust performance estimates
    - Learning curves diagnose fitting issues
3. **Evaluation Metrics**
    - Choose metrics appropriate for the problem
    - Classification: accuracy, precision, recall, F1, ROC-AUC
    - Regression: MSE, RMSE, MAE, R^2
4. **Regularization**
    - Ridge (L2): shrinks coefficients, keeps all features
    - Lasso (L1): feature selection via sparsity
    - Elastic Net: combines L1 and L2

Key Takeaways

Critical Principles

- **Generalization is the goal** - training performance is not enough
- **Avoid data leakage** - fit preprocessing only on training data
- **Use proper validation** - cross-validation for robust estimates
- **Test set is sacred** - evaluate on it only once at the end
- **Choose appropriate metrics** - align with business/research goals
- **Regularize when needed** - prevent overfitting proactively
- **Document everything** - ensure reproducibility

Next Steps

Practice model selection and evaluation on real datasets using cross-validation, regularization, and proper evaluation protocols.

Additional Resources

Textbooks

- Hastie, Tibshirani, Friedman - *The Elements of Statistical Learning*
- Bishop - *Pattern Recognition and Machine Learning*
- James et al. - *An Introduction to Statistical Learning*

Online Resources

- scikit-learn documentation: Model selection and evaluation
- Coursera: Machine Learning by Andrew Ng
- Fast.ai: Practical Deep Learning for Coders

Thank you!