Model Evaluation and Selection

Bias-Variance Tradeoff, Overfitting, Underfitting, and Model Complexity

Sarwan Ali

Department of Computer Science
Georgia State University

Model Evaluation & Selection

Today's Learning Journey

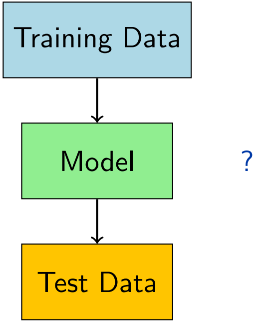
Why Model Evaluation Matters

Key Questions:

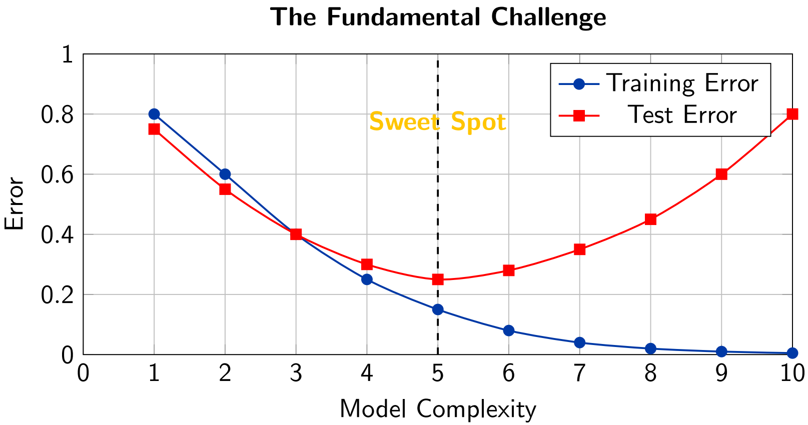
- How well does our model perform on unseen data?
- Is our model too simple or too complex?
- How do we choose between different models?
- What causes poor generalization?

The Goal:

Build models that **generalize well** to new, unseen data



Training vs. Test Performance



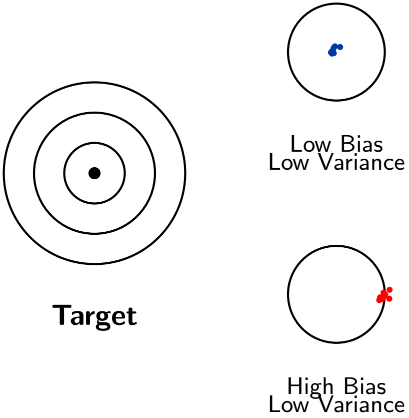
Understanding Bias and Variance

Bias:

- Error due to oversimplifying assumptions
- How far off is the average prediction?
- High bias = underfitting

Variance:

- Error due to sensitivity to training data
- How much do predictions vary?
- High variance = overfitting



Mathematical Formulation

For a prediction $\hat{f}(x)$ at point x, the expected test error decomposes as:

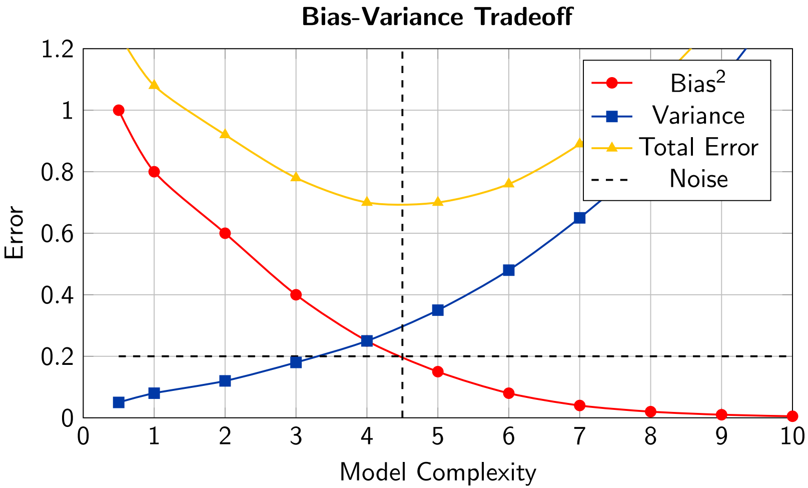
$$\text{Expected Test Error} = \text{Bias}^2 + \text{Variance} + \text{Noise} \tag{1}$$

$$E[(y - \hat{f}(x))^2] = \underbrace{[\text{Bias}(\hat{f}(x))]^2}_{\text{Underfitting}} + \underbrace{\text{Var}(\hat{f}(x))}_{\text{Overfitting}} + \underbrace{\sigma^2}_{\text{Irreducible}} \tag{2}$$

Where:

- $\text{Bias}(\hat{f}(x)) = E[\hat{f}(x)] - f(x)$ (systematic error)
- $\text{Var}(\hat{f}(x)) = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$ (variability)
- σ^2 is the irreducible error (noise in data)

The Tradeoff Visualization



Underfitting: Too Simple Models

Characteristics:

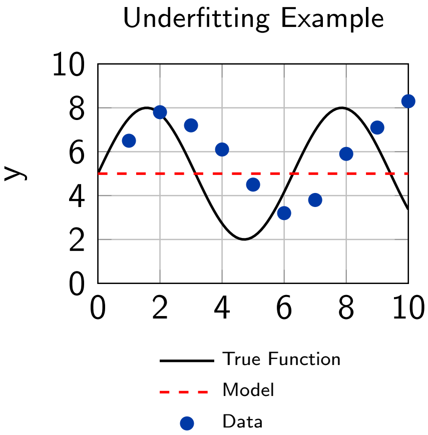
- High bias, low variance
- Poor performance on both training and test data
- Model is too simple to capture underlying patterns
- Systematic errors in predictions

Examples:

- Linear regression for non-linear data
- Decision tree with very few splits
- Neural network with too few neurons

Solutions:

- Increase model complexity
- Add more features
- Reduce regularization



Overfitting: Too Complex Models

Characteristics:

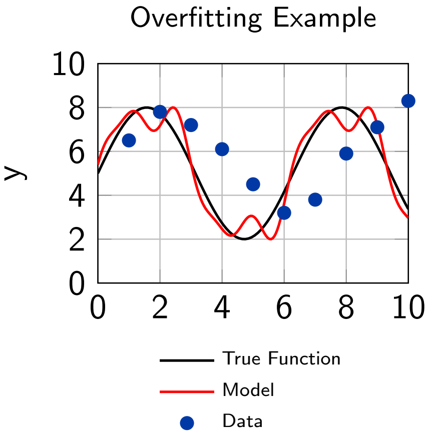
- Low bias, high variance
- Excellent training performance
- Poor test performance
- Model memorizes training data

Examples:

- High-degree polynomial regression
- Decision tree with many deep splits
- Neural network with too many parameters

Solutions:

- Reduce model complexity
- Add regularization
- Collect more training data
- Early stopping



Identifying Over/Underfitting

Increase Complexity

Decrease Complexity

Underfitting

Training Error: HIGH

Test Error: HIGH

Gap: SMALL

Good Fit

Training Error: LOW

Test Error: LOW

Gap: SMALL

Overfitting

Training Error: LOW

Test Error: HIGH

Gap: LARGE

Key Insight: The **gap** between training and test error is often more important than absolute error values!

What is Model Complexity?

Model complexity refers to the capacity of a model to fit diverse patterns in data.

Low Complexity:

- Few parameters
- Simple functional forms
- Strong assumptions
- Limited flexibility

Examples:

- Linear regression
- Naive Bayes
- Shallow decision trees

High Complexity:

- Many parameters
- Complex functional forms
- Fewer assumptions
- High flexibility

Examples:

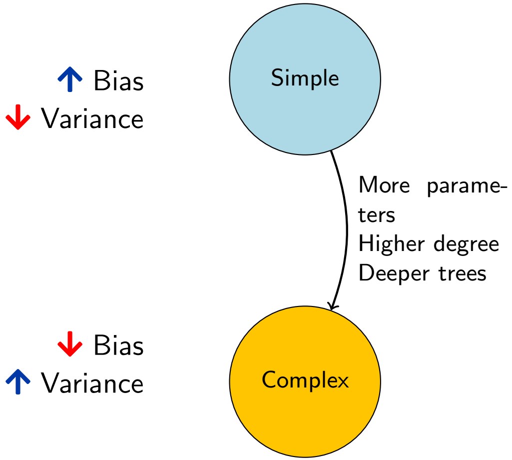
- Deep neural networks
- High-degree polynomials
- Deep decision trees

⚠ **Complexity is not just about number of parameters!**

Measuring Model Complexity

Common Measures:

1. **Number of Parameters**
    - Most intuitive measure
    - More parameters = more complexity
2. **VC Dimension**
    - Theoretical measure
    - Maximum points that can be shattered
3. **Rademacher Complexity**
    - Measures richness of function class
    - Based on random noise fitting
4. **Regularization Parameter**
    - Inverse relationship with complexity
    - Higher regularization = lower complexity



Complexity in Different Models

Model Type	**Complexity Controller**	**Effect**
Polynomial Regression	Degree of polynomial	Higher degree = more complex
Decision Trees	Max depth, min samples	Deeper/smaller splits = more complex
Neural Networks	# layers, # neurons	More layers/neurons = more complex
k-NN	Value of k	Smaller k = more complex
SVM	Regularization parameter C	Higher C = more complex
Ridge/Lasso	Regularization parameter λ	Smaller λ = more complex

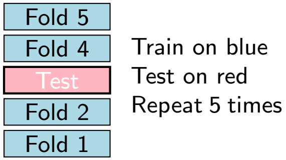
Key Insight: Different models have different ways to control complexity, but the bias-variance tradeoff applies universally!

Cross-Validation for Model Selection

Goal: Estimate how well our model generalizes to unseen data

k-Fold Cross-Validation:

1. Split data into k equal folds
2. For each fold:
   - Train on k-1 folds
   - Validate on remaining fold
3. Average validation scores



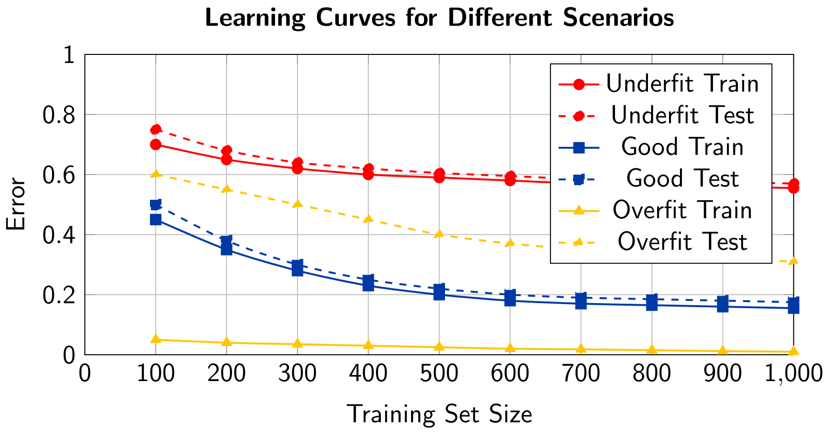
5-Fold CV

Benefits:

- Uses all data for both training and validation
- Reduces variance in performance estimates
- Helps detect overfitting

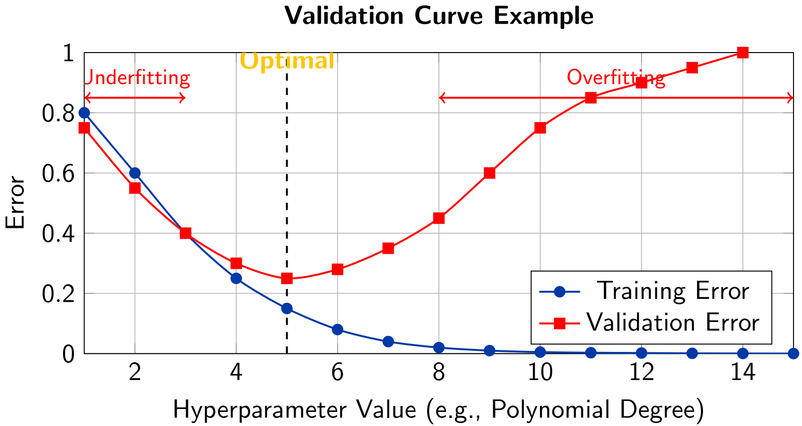
Learning Curves

Learning curves show performance vs. training set size



Validation Curves

Validation curves show performance vs. hyperparameter values



Use validation curves to: Select optimal hyperparameters, identify over/underfitting regions

Introduction to Regularization

Regularization: Adding a penalty term to prevent overfitting

Original Loss + Regularization Penalty = Total Loss

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \cdot \mathcal{R}(\theta)$$

Fit to data   Regularization strength   Penalty on complexity

Key Ideas:

- λ controls the bias-variance tradeoff
- Higher $\lambda \rightarrow$ simpler models (higher bias, lower variance)
- Lower $\lambda \rightarrow$ more complex models (lower bias, higher variance)

L1 and L2 Regularization

L2 Regularization (Ridge): $\mathcal{R}(\theta) = \sum_{i=1}^{p} \theta_i^2$

Properties:

- Shrinks coefficients toward zero
- Keeps all features
- Smooth penalty function
- Handles multicollinearity well

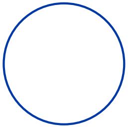
Effect: Coefficients become smaller but remain non-zero

L1 Regularization (Lasso): $\mathcal{R}(\theta) = \sum_{i=1}^{p} |\theta_i|$

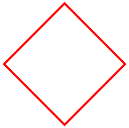
Properties:

- Can set coefficients exactly to zero
- Performs feature selection
- Non-smooth at zero
- Sparse solutions

Effect: Some coefficients become exactly zero



L2: Circle
$\theta_1^2 + \theta_2^2 \leq t$



L1: Diamond
$|\theta_1| + |\theta_2| \leq t$

Elastic Net: Best of Both Worlds

Elastic Net combines L1 and L2 regularization:

$$\mathcal{R}(\theta) = \alpha \sum_{i=1}^{p} |\theta_i| + (1-\alpha) \sum_{i=1}^{p} \theta_i^2$$

Hyperparameters:

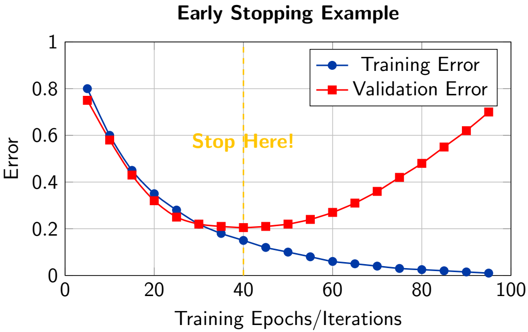
- λ: Overall regularization strength
- $\alpha \in [0, 1]$: Mix between L1 and L2
  - $\alpha = 0$: Pure L2 (Ridge)
  - $\alpha = 1$: Pure L1 (Lasso)
  - $0 < \alpha < 1$: Combination

Advantages:

- Feature selection like Lasso
- Stability like Ridge
- Handles correlated features better than Lasso alone

Early Stopping

Stop training when validation performance stops improving



Implementation:

- Monitor validation error during training
- Stop when validation error increases for several epochs
- Use patience parameter to avoid stopping too early

Classification Metrics

Confusion Matrix:

	Pred +	**Pred -**
Actual +	TP	FN
Actual -	FP	TN

Key Metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

When to Use:

- **Accuracy:** Balanced datasets
- **Precision:** When false positives are costly
- **Recall:** When false negatives are costly
- **F1-Score:** Imbalanced datasets

ROC-AUC:

- Area Under ROC Curve
- Plots True Positive Rate vs False Positive Rate
- Good for binary classification
- Range: [0, 1], higher is better

Precision-Recall AUC:

- Better for imbalanced datasets
- Focuses on positive class performance

Regression Metrics

Common Regression Metrics:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

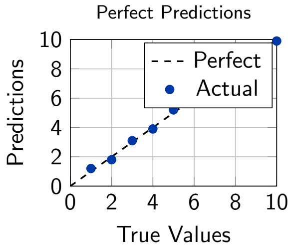
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{8}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{9}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{10}$$

Properties:

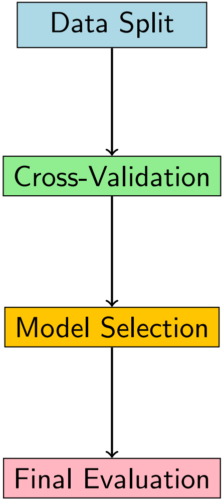
- MSE/RMSE: Penalize large errors more
- MAE: Robust to outliers
- R^2: Proportion of variance explained



Choosing Metrics:

- **RMSE:** When large errors matter more
- **MAE:** When all errors matter equally
- R^2**:** For model comparison

Model Selection Workflow



- Train/Validation/Test
- Typically 60/20/20 or 70/15/15

- k-fold CV on train+validation
- Tune hyperparameters
- Compare different models

- Choose best model
- Based on CV results
- Consider complexity vs performance

- Test on held-out set
- Report final performance
- NO more tuning allowed!

Detecting and Preventing Overfitting

Warning Signs:

- Large gap between train and validation error
- Training error continues decreasing while validation error increases
- Model performs much worse on new data
- Very complex model with little improvement

Detection Methods:

- Learning curves
- Validation curves
- Cross-validation
- Hold-out validation

Prevention Strategies:

- **More Data:** Collect additional training samples
- **Regularization:** L1, L2, or Elastic Net
- **Early Stopping:** Stop training early
- **Dropout:** For neural networks
- **Feature Selection:** Remove irrelevant features
- **Ensemble Methods:** Combine multiple models
- **Cross-Validation:** For model selection

⚠ **Remember:** Prevention is better than cure!

Handling Underfitting

Warning Signs:

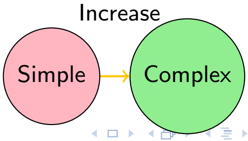
- Both training and validation errors are high
- Small gap between train and validation error
- Model performs poorly even on training data
- Learning curves plateau quickly

Common Causes:

- Model too simple for the data
- Insufficient features
- Over-regularization
- Poor feature engineering

Solutions:

- **Increase Complexity:** More parameters, deeper models
- **Feature Engineering:** Add polynomial features, interactions
- **Reduce Regularization:** Lower λ values
- **Different Model:** Try more flexible algorithms
- **Domain Knowledge:** Add relevant features
- **Data Preprocessing:** Better scaling, encoding



Best Practices Summary

Model Evaluation Best Practices

1. Data Management:

- Always keep a separate test set
- Never use test data for model selection
- Use stratified sampling for imbalanced data

2. Model Selection:

- Use cross-validation for hyperparameter tuning
- Compare multiple models systematically
- Consider computational constraints

3. Evaluation Strategy:

- Choose appropriate metrics for your problem
- Use multiple metrics to get complete picture
- Plot learning and validation curves

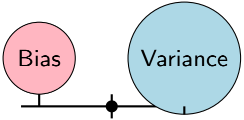
4. Overfitting Prevention:

- Start simple, then increase complexity
- Monitor training vs validation performance
- Use regularization techniques appropriately

Key Takeaways

The Big Picture:

1. **Bias-Variance Tradeoff** is fundamental
   - Every model has this tradeoff
   - Optimal complexity balances both
2. **Overfitting vs Underfitting**
   - Monitor training vs validation performance
   - Use appropriate techniques for each
3. **Model Selection** requires careful methodology
   - Cross-validation is your friend
   - Never peek at test data
4. **Regularization** helps control complexity
   - L1, L2, Elastic Net, Early Stopping
   - Choose based on your needs



Balance
is Key!

Remember:
"All models are wrong, but some are useful"
- George Box

Next Steps

What's Coming Next:

- **Ensemble Methods:** Combining multiple models
  - Bagging, Boosting, Stacking
  - Random Forests, Gradient Boosting
- **Advanced Regularization:** Beyond L1/L2
  - Dropout, Batch Normalization
  - Data Augmentation techniques
- **Model Interpretation:** Understanding model decisions
  - Feature importance, LIME, SHAP
  - Interpretability vs Performance tradeoffs

🚀 **Keep practicing with real datasets!** 🚀

Thank You!

Questions & Discussion

sali85@student.gsu.edu