# Ensemble Learning

## Beyond Single Models: Bias, Variance, and Model Combinations

## Part 1: Foundation Concepts

- Bias vs Variance Tradeoff
- How Regularization Affects This Tradeoff

## Part 2: Ensemble Methods

- Introduction to Ensemble Learning
- Random Forest (Bagging Paradigm)
- Gradient Boosting (Boosting Paradigm)

# Part 1: Bias-Variance Tradeoff

## The Fundamental Machine Learning Challenge

**Every ML model faces a crucial tradeoff:**

**Bias:** Error from oversimplifying assumptions

- Underfitting the true relationship
- High bias = systematic errors

**Variance:** Error from sensitivity to small changes in training data

- Overfitting to specific training examples
- High variance = inconsistent predictions

# Understanding Bias

## High Bias Models:

- Linear regression on non-linear data
- Shallow decision trees
- Simple assumptions about data relationships

## Characteristics:

- Poor performance on training data
- Poor performance on test data
- Consistent (but wrong) predictions
- **Underfitting problem**

**Example:** Using linear model for quadratic relationship

# Understanding Variance

## High Variance Models:

- Deep neural networks with little data
- Deep decision trees without pruning
- Complex models with many parameters

## Characteristics:

- Excellent performance on training data
- Poor performance on test data
- Predictions change dramatically with new training data
- **Overfitting problem**

# The Bias-Variance Decomposition

**Total Error = Bias² + Variance + Irreducible Error**

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \sigma^2$$

**Where:**

- **Bias²:** Systematic errors from model assumptions
- **Variance:** Sensitivity to training data variations
- **σ²:** Irreducible noise in the data

**Key Insight:** Usually cannot minimize both bias and variance simultaneously

# Model Complexity and the Tradeoff

**Simple Models (Low Complexity):**

- High Bias, Low Variance
- Consistent but potentially wrong predictions
- Examples: Linear regression, logistic regression

**Complex Models (High Complexity):**

- Low Bias, High Variance
- Accurate but inconsistent predictions
- Examples: Deep neural networks, unpruned decision trees

**Sweet Spot:** Balance between bias and variance for minimum total error

# How Regularization Affects Bias-Variance

## Regularization's Impact

**Regularization increases bias but decreases variance**

**L1/L2 Regularization Effect:**

- Forces simpler models (increases bias)
- Reduces sensitivity to training data (decreases variance)
- Often results in better generalization

**Hyperparameter C controls the tradeoff:**

- Small C: More regularization → Higher bias, Lower variance
- Large C: Less regularization → Lower bias, Higher variance

# Regularization in Practice

**Without Regularization:**

- Model fits training data very closely
- High variance, low bias
- Poor generalization

**With Optimal Regularization:**

- Model makes slightly biased predictions
- Much lower variance
- Better test performance

**Over-regularization:**

- Model becomes too simple , High bias, very low variance, Underfitting occurs

# Part 2: Ensemble Learning

## The Ensemble Philosophy

**Core Principle:** Instead of learning one super-accurate model, train many low-accuracy models and combine their predictions.

**Key Insight:** "The wisdom of crowds" - multiple weak learners can create a strong meta-model.

**Requirements for Success:**

1. Each weak model must be better than random guessing
2. Models should make different types of errors (diversity)
3. Effective combination strategy

# Why Ensemble Learning Works

**Individual Weak Models:**

- Fast to train and predict
- Usually shallow decision trees
- Each slightly better than random
- Make different mistakes

**Combined Strong Model:**

- Weighted voting aggregates predictions
- Good models agree on correct predictions
- Bad models disagree on incorrect predictions
- Majority vote or averaging leads to accuracy

# Weak Learners in Practice

**Most Common Weak Learner:** Shallow decision trees

**Characteristics:**

- Stop splitting after few iterations
- Not particularly accurate individually
- Fast training and prediction
- Each tree sees different aspects of data

**Why Trees?**

- Handle non-linear relationships
- Naturally handle mixed data types
- Easy to make diverse (different splits)

# Two Ensemble Paradigms

**Bagging (Bootstrap Aggregating): - Reduces variance**

- Create multiple copies of training data
- Train weak learners on different samples
- Combine predictions by averaging/voting
- Example: Random Forest

**Boosting: - Reduces bias**

- Train weak learners sequentially
- Each learner fixes errors of previous ones
- Combine with weighted voting
- Example: Gradient Boosting

# The Bagging Algorithm

**Step 1:** Create B random samples $S_b$ from training set

- Sample **with replacement** (bootstrap sampling)
- Each sample $|S_b| = N$ (same size as original)
- Each sample slightly different

**Step 2:** Train decision tree $f_b$ on each sample $S_b$

**Step 3:** Combine predictions:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} f_b(x) \quad \text{(regression)}$$

$$\hat{y} = \text{majority\_vote}(f_1(x), f_2(x), \ldots, f_B(x)) \quad \text{(classification)}$$

# Random Forest Enhancement

**Problem with Vanilla Bagging:** Trees become correlated

- Strong predictive features dominate splits
- Many trees use same features
- Correlation reduces ensemble benefit

**Random Forest Solution:** Feature randomness

- At each split, consider random subset of features
- Typically $\sqrt{p}$ features out of $p$ total features
- Forces trees to be different
- Reduces correlation between trees

# Why Random Forest Works

**Variance Reduction:** Multiple bootstrap samples reduce overfitting

- Averages out artifacts in any single sample
- Noise and outliers have less impact
- Model becomes more stable

**Feature Diversity:** Random feature selection

- Prevents feature dominance
- Ensures tree diversity
- Different trees capture different patterns

**Key Insight:** Uncorrelated predictors are essential for ensemble success

# Random Forest Hyperparameters

**Critical Parameters:**

1. **Number of trees (B):**
    - More trees → better performance (diminishing returns)
    - Typical range: 100-1000 trees
2. **Feature subset size:**
    - Regression: $p/3$ features
    - Classification: $\sqrt{p}$ features
    - Can tune based on validation performance
3. **Tree depth:** Usually left deep (individual trees can overfit)

# Random Forest Advantages

**Why Random Forest is Popular:**

- **Robust:** Handles outliers and noise well
- **Fast:** Embarrassingly parallel training
- **No overfitting:** More trees rarely hurt performance
- **Feature importance:** Built-in feature ranking
- **Mixed data types:** Handles categorical and numerical features
- **Missing values:** Can handle missing data naturally

**Main Limitation:** Less interpretable than single decision tree

# 7.5.2 Gradient Boosting

## The Boosting Philosophy

**Sequential Learning:** Each model fixes errors of previous models

**Key Difference from Bagging:**

- Bagging: Models trained independently
- Boosting: Models trained sequentially, each learning from previous mistakes

**Focus:** Reduce bias (underfitting) rather than variance

# Gradient Boosting for Regression

**Step 1:** Start with simple baseline

$$f_0(x) = \frac{1}{N}\sum_{i=1}^{N} y_i \quad (\text{mean of targets})$$

**Step 2:** Compute residuals (errors)

$$\hat{y}_i = y_i - f(x_i)$$

**Step 3:** Train new tree $f_1$ to predict residuals

**Step 4:** Update model

$$f = f_0 + \alpha f_1$$

**Step 5:** Repeat until M trees added

$$f = f_0 + \alpha f_1 + \alpha f_2 + \ldots + \alpha f_M$$

# Intuition Behind Gradient Boosting

**What's Happening:**

1. Current model makes predictions
2. Calculate where model is wrong (residuals)
3. Train new model to fix these specific errors
4. Add correction to ensemble with learning rate $\alpha$

**Learning Rate $\alpha$:**

- Controls how much each tree contributes
- Small $\alpha$: Conservative updates, needs more trees
- Large $\alpha$: Aggressive updates, risk of overfitting

**Analogy:** Student learning from mistakes on practice exams

# Why "Gradient" Boosting?

**Connection to Gradient Descent:**

**Gradient Descent:** Move parameters to minimize cost function

- Use gradient to find direction
- Use learning rate for step size

**Gradient Boosting:** Add models to minimize prediction error

- Use residuals as "gradient proxy"
- Use learning rate for contribution weight

**Key Insight:** Residuals show direction to improve predictions, just like gradients show direction to improve parameters

# Gradient Boosting Hyperparameters

**Three Critical Parameters:**

1. **Number of trees (M):**
    - More trees can improve accuracy
    - Risk of overfitting with too many
2. **Learning rate ($\alpha$):**
    - Smaller rates need more trees but often generalize better
    - Typical range: 0.01 - 0.3
3. **Tree depth:**
    - Shallow trees (depth 3-8) often work best
    - Deeper trees → slower training/prediction

# Gradient Boosting for Classification

## Binary Classification Setup:

$$P(y = 1|x) = \frac{1}{1 + e^{-f(x)}}$$

where $f(x) = \sum_{m=1}^{M} f_m(x)$

## Algorithm Differences:

- Initialize with $f_0 = \ln\left(\frac{p}{1-p}\right)$ where $p = \frac{1}{N}\sum_{i=1}^{N} y_i$
- Compute gradients: $g_i = \frac{\partial L_f}{\partial f}$ for each example
- Train tree on gradients instead of residuals
- Use line search to find optimal step size

# Random Forest vs Gradient Boosting

**Random Forest:**

- **Parallel training** (fast)
- **Reduces variance**
- Harder to overfit, Good default choice, Less sensitive to hyperparameters

**Gradient Boosting:**

- **Sequential training** (slower)
- **Reduces bias**
- Can overfit with too many trees, Often higher accuracy, More hyperparameter tuning needed

**Rule of thumb:** Try Random Forest first, use Gradient Boosting for maximum accuracy

# Ensemble Learning in Practice

**Modern Implementations:**

- **Random Forest:** scikit-learn RandomForestClassifier/Regressor
- **Gradient Boosting:** XGBoost, LightGBM, CatBoost

**When to Use Ensembles:**

- Tabular data competitions (almost always win), When accuracy is critical
- When you have sufficient training data, When interpretability is not primary concern

**Limitations:**

- More complex than single models
- Harder to interpret
- Longer training times (especially boosting)

# Key Takeaways

**Bias-Variance Tradeoff:**

- Fundamental challenge in ML
- Regularization increases bias, decreases variance
- Sweet spot depends on problem and data size

**Ensemble Learning:**

- Combines weak learners into strong meta-model
- Random Forest reduces variance through bagging
- Gradient Boosting reduces bias through sequential error correction
- Often achieves best predictive performance

**Practical Wisdom:** Master these ensemble methods - they're workhorses of modern ML

# Important Clarifications: A Deeper Look

**Regularization Effects: ✓ Correctly Understood**

- Regularization reduces variance by constraining model complexity
- Prevents overfitting to training data noise
- May slightly increase bias but improves generalization

**Ensemble Effects: More Nuanced Than Initially Presented**

# Important Clarifications: A Deeper Look

**Bagging Ensembles (Random Forest):**

- **Primary effect:** Reduce variance through averaging
- **Bias effect:** Keeps bias roughly the same
- **Best for:** High variance (overfitting) models

**Boosting Ensembles (Gradient Boosting):**

- **Primary effect:** Reduce bias through sequential error correction
- **Secondary effect:** May slightly increase variance
- **Best for:** High bias (underfitting) models

# The Complete Bias-Variance Picture

**Model Complexity Effects:**

- **Increasing complexity:** Decreases bias ↓, increases variance ↑
- **Decreasing complexity:** Increases bias ↑, decreases variance ↓

**Different Solutions for Different Problems:**

**High Bias Models (Underfit):**

- **Solution:** Boosting ensembles, increase model complexity
- **Examples:** Linear regression on complex data, shallow trees

**High Variance Models (Overfit):**

- **Solution:** Regularization, bagging ensembles, reduce complexity
- **Examples:** Deep trees, complex neural networks

**For Underfitting (High Bias):**

1. Try Gradient Boosting
2. Increase model complexity
3. Add more features
4. Reduce regularization

**For Overfitting (High Variance):**

1. Try Random Forest
2. Add regularization
3. Reduce model complexity
4. Collect more training data

**Key Insight:** Choose your ensemble method based on whether your base models have high bias or high variance