Random Forest Hyperparameters

Jerome Chou

May 3, 2025

Outline

Key Random Forest Hyperparameters (1/2)

- **n_estimators**: Number of decision trees
  - Typical range: 50-500, depending on dataset size and complexity
  - Too few: underfitting; Too many: diminishing returns
- **max_depth**: Maximum depth of each tree
  - Controls tree complexity and prevents overfitting
  - Options: integer or None (unlimited)
  - Typical range: 10-100
- **min_samples_split**: Minimum samples required to split a node
  - Controls how strictly nodes are split
  - Options: integer or float (percentage)
  - Larger values reduce overfitting

Key Random Forest Hyperparameters (2/2)

- **min_samples_leaf**: Minimum samples required in leaf nodes
  - Ensures each leaf has certain number of samples
  - Options: integer or float (percentage)
  - Typical range: 1-10
- **max_features**: Number of features to consider for each split
  - Options: "sqrt", "log2", integer, float, None
  - Classification recommendation: "sqrt"
- **criterion**: Function to measure split quality
  - Classification: "gini" (default), "entropy"
  - Regression: "squared_error" (default), "absolute_error", "friedman_mse"

Additional Hyperparameters

- **bootstrap**: Whether to use bootstrap sampling
  - True: Use bootstrap samples to build trees
  - False: Use all data for each tree
- **random_state**: Random seed for reproducibility
- **oob_score**: Whether to use out-of-bag samples for evaluation
- **class_weight**: Weights for handling imbalanced data
  - "balanced": Weights inversely proportional to class frequencies
  - Dictionary: Manually specify weights per class
- **max_samples**: Number of samples to draw from training data
- **warm_start**: Whether to reuse previous trees when adding more

Dimensions of Model Complexity

Structural Complexity:

- Number of trees
- Tree depth
- Total node count
- Leaf node count

Computational Complexity:

- Training/prediction time
- Model size (memory)
- Feature usage statistics

Statistical Complexity:

- Effective parameter count
- Model variance
- Training-validation performance gap
- Minimum Description Length (MDL)

Hyperparameters and Complexity Relationship

Hyperparameter	Relationship	Effect on Complexity
n_estimators	Linear	$\uparrow$ More trees $\rightarrow$ $\uparrow$ Complexity
max_depth	Exponential	$\uparrow$ Deeper trees $\rightarrow$ $\uparrow$ Complexity
min_samples_split	Inverse	$\downarrow$ Lower threshold $\rightarrow$ $\uparrow$ Complexity
min_samples_leaf	Inverse	$\downarrow$ Lower threshold $\rightarrow$ $\uparrow$ Complexity
max_features	Direct	$\uparrow$ More features $\rightarrow$ $\uparrow$ Complexity

Complexity Index Calculation

Complexity Index Formula:

$$C = n\_estimators \times 2^{max\_depth} \times \frac{max\_features}{total\_features} \tag{1}$$

Intuition:

- Trees contribute linearly to complexity
- Depth has exponential impact (each level doubles potential splits)
- Feature ratio captures feature selection impact

Example:

- Low complexity:
  $C = 100 \times 2^{10} \times 0.3 = 100 \times 1024 \times 0.3 \approx 30,720$
- High complexity:
  $C = 500 \times 2^{20} \times 1.0 = 500 \times 1,048,576 \times 1.0 \approx 524,288,000$

Parameter Combination Analysis

Key Insight: Interactions between parameters affect complexity more than individual settings

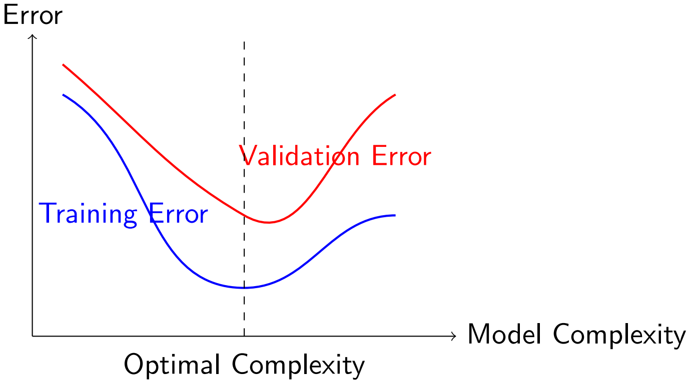
max_depth $\times$ **min_samples_leaf** relationship:

- These parameters have complementary effects:
  - max_depth: Limits vertical growth
  - min_samples_leaf: Limits horizontal splitting
- Their product forms a compound complexity measure
- **Smaller product** $\rightarrow$ **Higher model complexity**

Why? Small products allow trees to either:

- Grow very deep with few samples per leaf, or
- Create many fine-grained decisions with minimal data support

Complexity Curves



Using Complexity Curves:

- Fix all parameters except one (e.g., max_depth)
- Plot training and validation errors against this parameter
- Observe where validation error starts increasing
- This inflection point indicates optimal complexity balance

Practical Complexity Tuning Strategy

1. **Start simple:** Begin with low complexity
   - Fewer trees (n_estimators = 100)
   - Limited depth (max_depth = 10)
   - Conservative leaf size (min_samples_leaf = 5)
2. **Systematic increase:** Gradually increase complexity
   - Monitor training and validation performance
   - Track parameter products (max_depth × min_samples_leaf)
   - Use grid search or random search for multiple parameters
3. **Stop condition:** When validation performance plateaus or declines
4. **Final check:** Verify with cross-validation on best candidates

Takeaways

- **Random Forest complexity is multidimensional:**
  - Structural (trees, depth, nodes)
  - Computational (time, memory)
  - Statistical (variance, generalization)
- **Key hyperparameters affect complexity differently:**
  - n_estimators: Linear relationship
  - max_depth: Exponential relationship
  - min_samples_leaf/split: Inverse relationship
- **Parameter interactions matter:**
  - Product of max_depth × min_samples_leaf is particularly important
  - Smaller products create more complex, potentially overfitted models
- **Complexity should be systematically tuned:**
  - Start simple and increase complexity gradually
  - Monitor validation performance for optimal balance