Comprehensive Review: Random Forests

Master's Level Data Science

Contents

1 Introduction

This review synthesizes the lecture slides (`ensemble-4.pdf`) and the audio transcript (`RandomForests.txt`) on random forests. We cover motivation, the random forest algorithm, illustrative diagrams, a worked example, empirical results on the Covertype dataset, practical guidelines, and future extensions.

2 Motivation: From Tree to Forest

Decision trees are expressive but prone to overfitting beyond a certain size. Boosting mitigates this by sequentially reweighting examples, but it is inherently sequential and cannot be parallelized efficiently. Random forests address both issues by building many trees in parallel with injected randomness to ensure diversity and then aggregating via majority vote.

3 Random Forest Algorithm

Given a dataset S of n labeled points:

1. **For** $t = 1, \ldots, T$ (in parallel):
   (a) Draw a *bootstrap sample* S_t of size n' (typically $n' = n$) by sampling with replacement from S.
   (b) Grow a decision tree h_t on S_t:
      - At each node, select a random subset of k features (often $k = \sqrt{d}$ or $d/2$).
      - Choose the best split among those k features.
      - Continue until a stopping criterion (e.g. max depth or purity) is met.
2. **Output** the ensemble classifier:

$$H(x) \;=\; \text{majority\_vote}\{\, h_t(x)\}_{t=1}^{T}.$$

This parallel procedure yields diverse trees whose errors tend to be uncorrelated, reducing variance in aggregation:contentReference[oaicite:1]index=1.

4 Geometric Illustration

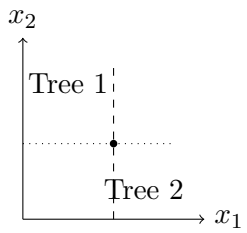

Figure 1: Two example trees with different splits due to random feature selection.

5 Worked Example

We demonstrate a random forest on a synthetic binary classification dataset.

5.1 Data Generation and Preprocessing

```
import numpy as np
from sklearn.datasets import make_classification
X, y = make_classification(
    n_samples=1000, n_features=10,
    n_informative=5, n_redundant=2,
    random_state=42
)
```

5.2 Model Training

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(
    n_estimators=100,
    max_features='sqrt',
    max_depth=None,
    bootstrap=True,
    random_state=42
)
clf.fit(X, y)
```

5.3 Model Evaluation

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, X, y, cv=5, scoring='accuracy')
print("CV accuracy: %.2f +/- %.2f" % (scores.mean(), scores.std()))
```

6 Empirical Results: Covertype Data

On the Covertype dataset (49,514 train / 445,627 test), reported errors are:contentReference[oaicite:2]index=2:

- Single decision tree (depth 20): test error 12.6%.
- Boosted trees (10 trees, depth 20): test error 8.7%.
- Random forest (10 trees, 50% features, depth 40): test error 8.8%.

7 Interpretation & Guidelines

- **Variance Reduction**: Aggregating uncorrelated trees lowers variance without increasing bias.
- **Feature Subsampling**: Random feature selection ensures diversity and prevents dominant features from being used in all trees.
- **Parallelism**: Trees can be trained in parallel, speeding up training.
- **Hyperparameters**: Tune T, k (features per split), n' (bootstrap size), and tree depth via cross-validation.

8 Future Directions / Extensions

- **Out-of-Bag Error**: Use OOB samples to estimate generalization error without a separate validation set.
- **Feature Importance**: Compute importance scores via permutation or mean decrease in impurity.
- **Extremely Randomized Trees**: Use random thresholds in addition to random features.

- **Hybrid Ensembles**: Combine random forests with boosting or other learners for enhanced performance.