Comprehensive Review: Overfitting in Decision Trees

Master's Level Data Science

Contents

1 Introduction

This review synthesizes the lecture slides (`dtree-2.pdf`) and audio transcript (`Overfitting.txt`) on overfitting in decision trees. We discuss why trees overfit, how to detect it, and strategies to prevent it, including pruning.

2 Overfitting in Decision Trees

Decision trees can fit any dataset perfectly by growing until each leaf contains a single example. While this drives training error to zero, it often captures noise and outliers, increasing true (generalization) error.

2.1 Illustrative Example

Consider a binary dataset in $\mathbb{R}^2$ with two classes (red circles, blue stars).

- *Shallow tree (2 splits)*: misclassifies one red point, simple partition.
- *Deep tree (full purity)*: zero training error, many splits to isolate outliers.

The deeper tree fits noise; the simpler tree may generalize better.

3 Error vs. Model Complexity

As we increase the number of internal nodes n:

$$E_{\text{train}}(n) \searrow 0, \quad E_{\text{true}}(n) \begin{cases} \searrow & n \leq n^* \\ \nearrow & n > n^* \end{cases}$$

where n^* is the complexity at which overfitting begins.

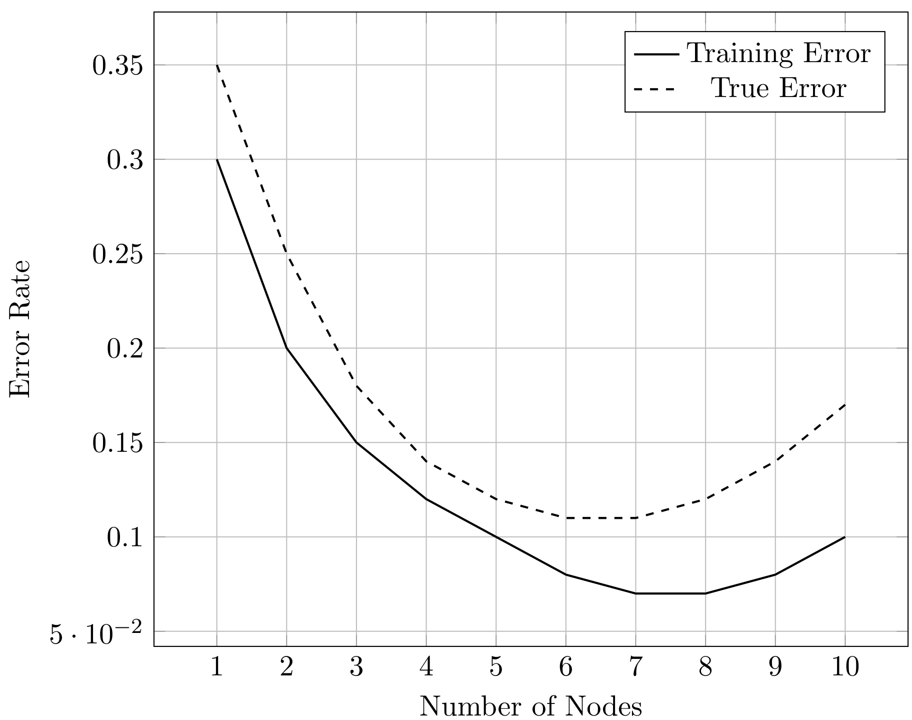

Figure 1: Training vs. true error as tree complexity increases.

4 Properties of Decision Trees

- **Expressive**: handle real, Boolean, categorical data.
- **Multi-class**: inherently support any number of classes.
- **Universal approximator**: can fit any finite dataset.
- **Interpretability**: produce human-readable rules.

Their expressivity leads to high overfitting risk.

5 Stopping Criteria and Pruning

5.1 Common Stop Rules

1. *Pure leaves*: stop when each leaf has one class.
2. *Max size*: limit number of nodes or depth.
3. *Uncertainty threshold*: stop when leaf impurity (e.g. Gini) $\leq \epsilon$.

Pure leaves eliminate training error but overfit; size or impurity thresholds require tuning.

5.2 Cost-Complexity Pruning

1. **Grow full tree** until purity.
2. **Generate pruned subtrees**: consider all ways to collapse internal nodes.
3. **Select best subtree** by lowest error on a validation set.
4. Efficient algorithms (e.g. weakest link pruning) find optimal subtree without exhaustive search.

6 Algorithm Summary

1. **Initialize** with root node containing all data.
2. **Grow**: repeatedly split leaves by maximizing impurity reduction.
3. **Stop**: when leaves are pure (for full growth).
4. **Prune**: using validation data, collapse branches to minimize validation error.

7 Geometric Illustrations

7.1 True vs. Overfit Boundary

8 Worked Example

8.1 Data and Model

Generate toy data and train two trees of different depths.

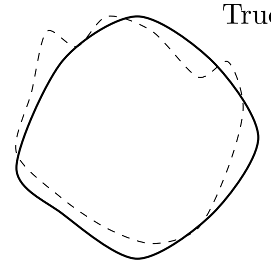

Figure 2: True decision boundary vs. overfit tree boundary.

```
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
X,y = make_classification(
  n_samples=300,n_features=2,n_informative=2,
  n_redundant=0,random_state=0
)
Xtr,Xte,ytr,yte = train_test_split(
  X,y,test_size=0.3,random_state=0
)
# Shallow tree
clf1 = DecisionTreeClassifier(max_depth=2, random_state=0)
clf1.fit(Xtr,ytr)
# Deep tree
clf2 = DecisionTreeClassifier(max_depth=None, random_state=0)
clf2.fit(Xtr,ytr)
```

8.2 Evaluation

```
from sklearn.metrics import accuracy_score
print("Shallow train/test:",
  accuracy_score(ytr, clf1.predict(Xtr)),
  accuracy_score(yte, clf1.predict(Xte)))
print("Deep train/test:",
  accuracy_score(ytr, clf2.predict(Xtr)),
  accuracy_score(yte, clf2.predict(Xte)))
```

9 Empirical Analysis

Typical results:

Model	Train Acc	Test Acc
Depth = 2	0.85	0.80
Full depth	1.00	0. seventy

10 Interpretation & Guidelines

- **Bias–Variance**: shallow trees: high bias, low variance; deep trees: low bias, high variance.
- **Validation**: use held-out data to choose tree size.

- **Regularization**: limit depth, require min samples per leaf.
- **Interpretability**: simpler trees yield clearer rules.

11 Future Directions / Extensions

- **Ensembles**: Random Forests, Gradient Boosting reduce variance.
- **Oblique Trees**: splits on linear combinations of features.
- **Cost-Sensitive Pruning**: incorporate misclassification costs.
- **Online Learning**: update trees on streaming data.