Machine Learning

| How to avoid Overfitting?

Prof. Dr. Mostafa Elhosseini

https://youtube.com/ElhosseiniAcademy

Agenda

- Parametric models
- Nonparametric models
- When to stop building the tree
- Regularization hyperparameters
- Pruning

Parametric Model

- Parametric models, such as **linear regression**, have a fixed number of parameters regardless of the amount or complexity of data.
  - For example, a linear model with two features always has three parameters: two coefficients and an intercept.

- This constraint in the number of parameters limits the model's flexibility, meaning it can't fit every nuance in the data.

- This **restriction** helps **prevent overfitting** but can **lead to underfitting**

Non-Parametric Model

- The structure (number of nodes, branches, and splits) is created dynamically based on the training data.
- Because decision trees grow in response to the data, the model can become very complex if left unconstrained. This means that it can fit the training data very closely, **often capturing noise** rather than meaningful patterns, **resulting in overfitting**.
- Such a model is often called a nonparametric model, **not because it does not have any parameters** (it often has a lot) but because the number of parameters is **not determined prior to training**, so the model structure is free to stick closely to the data

Parametric Vs. Non-Parametric Models

- **Nonparametric** models (like **decision trees**) are highly flexible but prone to overfitting because they adapt closely to the data structure.

- **Parametric** models (like **linear models**) have limited flexibility with a fixed number of parameters, making them less prone to overfitting but sometimes too rigid to capture complex relationships.

Non-Parametric Model

- To **avoid overfitting** the training data, you need to **restrict** the decision tree's freedom during training. As you know by now, this is called **regularization**.

- The regularization hyperparameters depend on the algorithm used, but generally you can **at least** restrict the **maximum depth** of the decision tree

When to stop building the tree

- Don’t split a node if the change in accuracy, Gini index, or entropy is below some threshold.
- Don’t split a node if it has less than a certain number of samples.
- Split a node only if both of the resulting leaves contain at least a certain number of samples.
- Stop building the tree after you reach a certain depth.

When to stop building the tree

All of these stopping conditions require a **hyperparameter**

- The minimum amount of change in accuracy (or Gini index, or entropy)
- The minimum number of samples that a node must have to split it
- The minimum number of samples allowed in a leaf node
- The maximum depth of the tree

Non-Parametric Model

- The DecisionTreeClassifier class has a few parameters that restrict the shape of the decision tree
- **max_depth**: The default value is None, which means unlimited. Reducing max_depth will regularize the model and thus reduce the risk of overfitting.
- **max_features**: Maximum number of features that are evaluated for splitting at each node
- **max_leaf_nodes**: Maximum number of leaf nodes

Non-Parametric Model

- **min_samples_split**: Minimum number of samples a node must have before it can be split
- **min_samples_leaf**: Minimum number of samples a leaf node must have to be created
- **min_weight_fraction_leaf**: Same as min_samples_leaf but expressed as a fraction of the total number of weighted instances
- Increasing min_* hyperparameters or reducing max_* hyperparameters will regularize the model.

Pruning

- Pruning is a **technique** used in decision trees to **reduce overfitting** and improve generalization by removing branches (or sub-trees) that do not contribute significantly to the predictive power of the model.
- By **eliminating** these less informative or overly specific branches, pruning helps the tree focus on the most meaningful patterns in the data rather than capturing noise.
- A node is considered unnecessary if the additional splits it introduces (through its children) do not provide a statistically significant improvement in purity.

Pruning

- Once the tree is fully grown, the algorithm starts removing branches or nodes that don’t add much value to the prediction process.
- A node is considered "unnecessary" if it doesn’t provide a significant improvement in purity (how well it separates the classes). For example, if removing it doesn’t change the prediction accuracy much.
- To decide if a node is necessary, the algorithm uses a statistical test called the chi-squared test. This test checks whether the improvement that a node provides is likely meaningful or just due to random chance.

Pruning

- The test result gives a p-value, which is the probability that the improvement from the node is due to random chance.
- If the p-value is higher than a set threshold (like 5%), it suggests that the node's improvement is probably due to chance, so the node is pruned (removed).This threshold (often set to 5%) is a hyperparameter that can be adjusted depending on how aggressively you want to prune the tree.

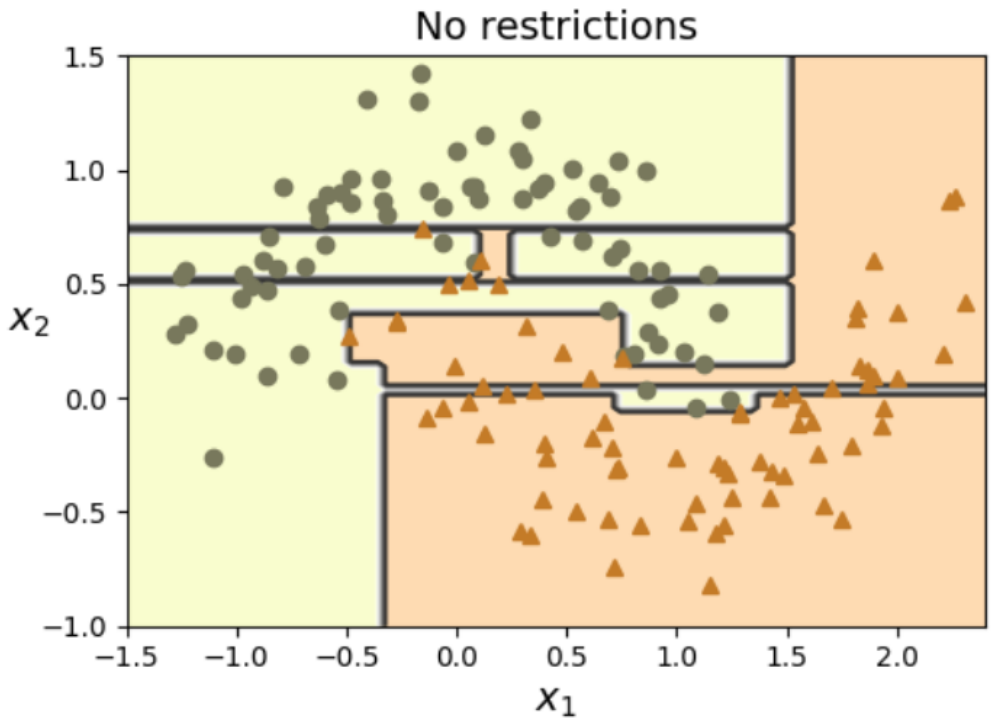
Pruning

- The moons dataset

```
from sklearn.datasets import make_moons

X_moons, y_moons = make_moons(n_samples=150, noise=0.2, random_state=42)

tree_clf1 = DecisionTreeClassifier(random_state=42)
tree_clf2 = DecisionTreeClassifier(min_samples_leaf=5, random_state=42)
tree_clf1.fit(X_moons, y_moons)
tree_clf2.fit(X_moons, y_moons)
```



Pruning

- The unregularized model on the left is clearly overfitting, and the regularized model on the right will probably generalize better