DECISION TREES

Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

DECISION TREE INDUCTION

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART – Classification and Regression Trees
  - ID3, C4.5
  - SLIQ,SPRINT

IS IT CLASSIFICATION OR REGRESSION?

There are two main types of Decision Trees:

1. Classification Trees (Yes/No types)

- The outcome is a variable like ‘spam’ or ‘not spam’. Here the decision variable is categorical/discrete.
- Such a tree is built through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches.

2. Regression Trees (Continuous data types)

  - Decision trees where the target variable can take continuous values (typically real numbers).
  - Returns average value at every node

DECISION TREE CLASSIFIER – WHAT IS IT?

- Used for Classification:
- Returns probability scores of class membership
  - Well-calibrated, like logistic regression
  - Assigns labels based on highest scoring class
  - Some Decision Tree algorithms return simply the mostly class
- Input variables can be continuous or discrete
- Output:
  - A tree that describes the decision flow.
  - Leaf nodes return either a probability score or simply a classification (label).
  - Trees can be converted to a set of "decision rules".
    - If/Then statements

IDENTIFY…..

Decision Tree consists of :

- Nodes : Test for the value of a certain attribute.
- Edges/ Branch : Correspond to the outcome of a test and connect to the next node or leaf.
- Leaf nodes : Terminal nodes that predict the outcome (represent class labels or class distribution).

- Internal Nodes?
- Leaf Nodes?
- Parents?
- Children?

TERMINOLOGIES OF THE DECISION TREE

- Root node (forms a class label),
- Decision nodes(sub-nodes),
- Terminal node (do not split further).
- The unique concept behind this machine learning approach is they classify the given data into classes that form yes or no flow (if-else approach) and represents the results in a tree structure.

TREE INDUCTION

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

BASIC DIVIDE-AND-CONQUER ALGORITHM :

1. Select a test for root node. Create branch for each possible outcome of the test.
2. Split instances into subsets. One for each branch extending from the node.
3. Repeat recursively for each branch, using only instances that reach the branch.
4. Stop recursion for a branch if all its instances have the same class.

TREE INDUCTION

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

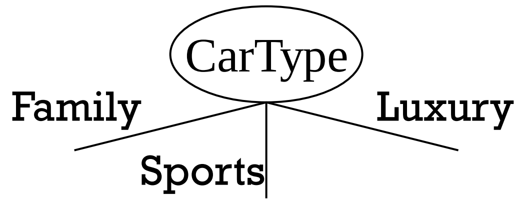
HOW TO SPECIFY TEST CONDITION?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

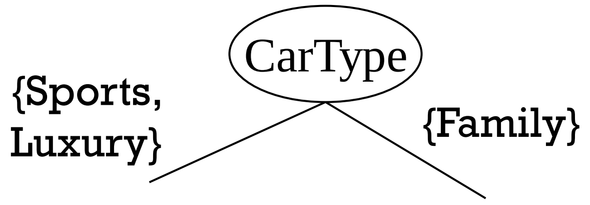
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

SPLITTING BASED ON NOMINAL ATTRIBUTES

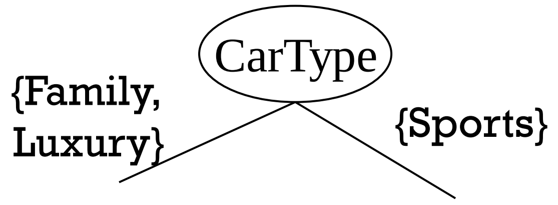
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.

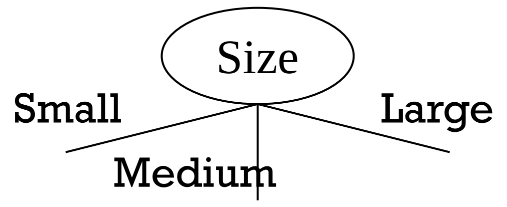


OR

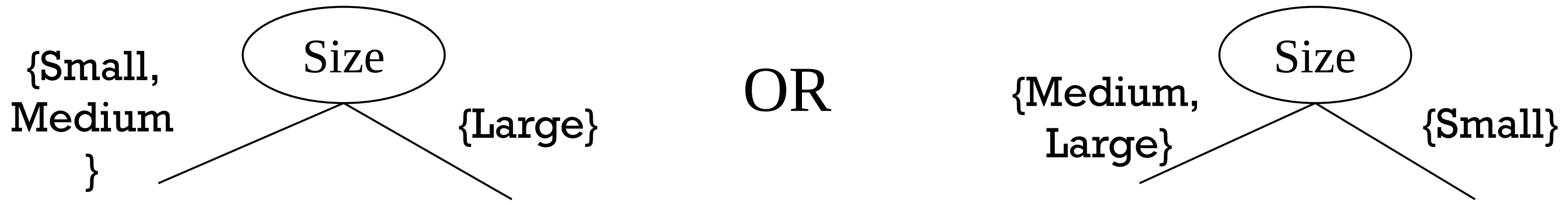


SPLITTING BASED ON ORDINAL ATTRIBUTES

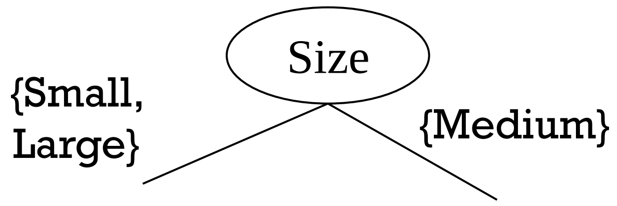
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.



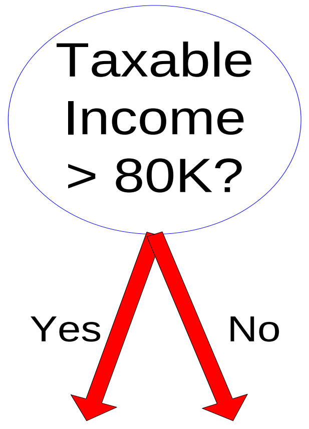
- What about this split?



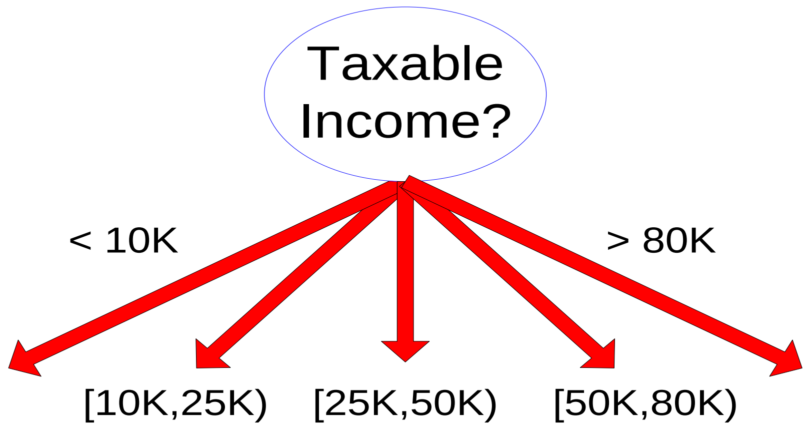
SPLITTING BASED ON CONTINUOUS ATTRIBUTES

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

SPLITTING BASED ON CONTINUOUS ATTRIBUTES



(i) Binary split



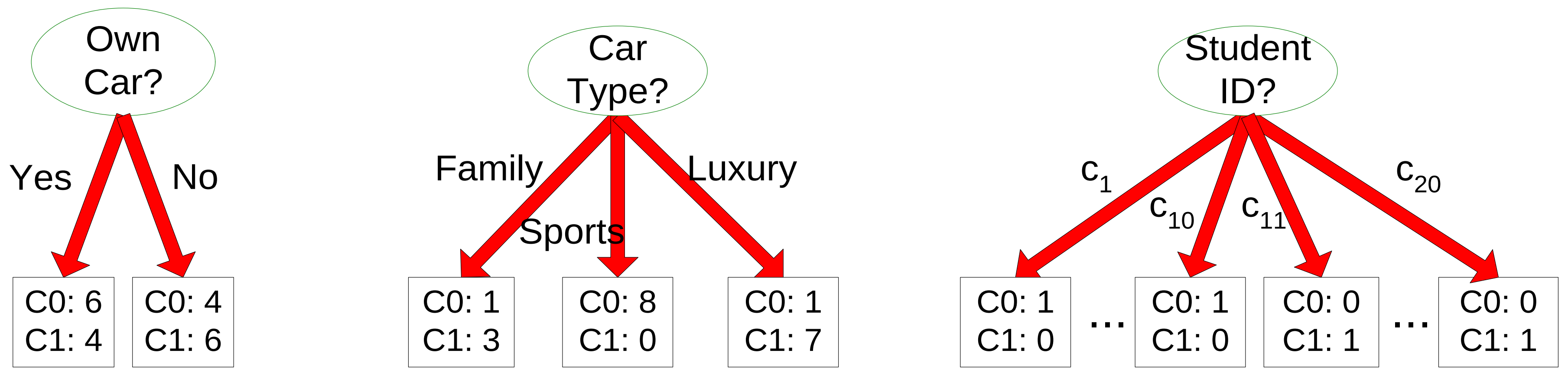
(ii) Multi-way split

TREE INDUCTION

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

HOW TO DETERMINE THE BEST SPLIT

**Before Splitting: 10 records of class 0,
10 records of class 1**



Which test condition is the best?

HOW TO DETERMINE THE BEST SPLIT

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5	C0: 9
C1: 5	C1: 1

Non-homogeneous,

High degree of impurity

Homogeneous,

Low degree of impurity

MEASURES OF NODE IMPURITY

- Gini Index
- Entropy
- Misclassification error

MEASURE OF IMPURITY: GINI

- Gini index gives an idea of how good a split is by how mixed the response classes are in the groups created by the split.
- Favors larger partitions.
- Uses squared proportion of classes.
- Perfectly classified split, Gini Index would be zero.
- Perfectly random split, Gini Index would be one.
- For a 2-class split, evenly distributed into classes would be 1 – (1/# Classes)=0.5.
- You want a variable split that has a low Gini Index.

GINI IMPURITY

- Measures how ‘pure’ our splits are. If a split results in one class being more predominant than another, e.g. 80% of class A and 20% of class B, this means that the split is 80% pure.
- The algorithm iteratively tries to find percentages like these of independent values, which produce homogenous classes.
- Do not confuse Gini impurity with Gini coefficient (also called Gini index), which is a popular econometric measure of inequality).

EXAMPLES FOR COMPUTING GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Gini = 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444

TREE INDUCTION

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

STOPPING CRITERIA FOR TREE INDUCTION

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (topic not apart of this course)

ADVANTAGES OF DECISION TREE BASED CLASSIFICATION (CART)

- Simple to understand, interpret, visualize.
- Decision trees *implicitly perform variable screening or feature selection.*
- Can *handle both numerical and categorical data*. Can also *handle multi-output problems.*
- Decision trees require relatively *little effort from users for data preparation.*
- *Nonlinear relationships between parameters do not affect tree performance.*

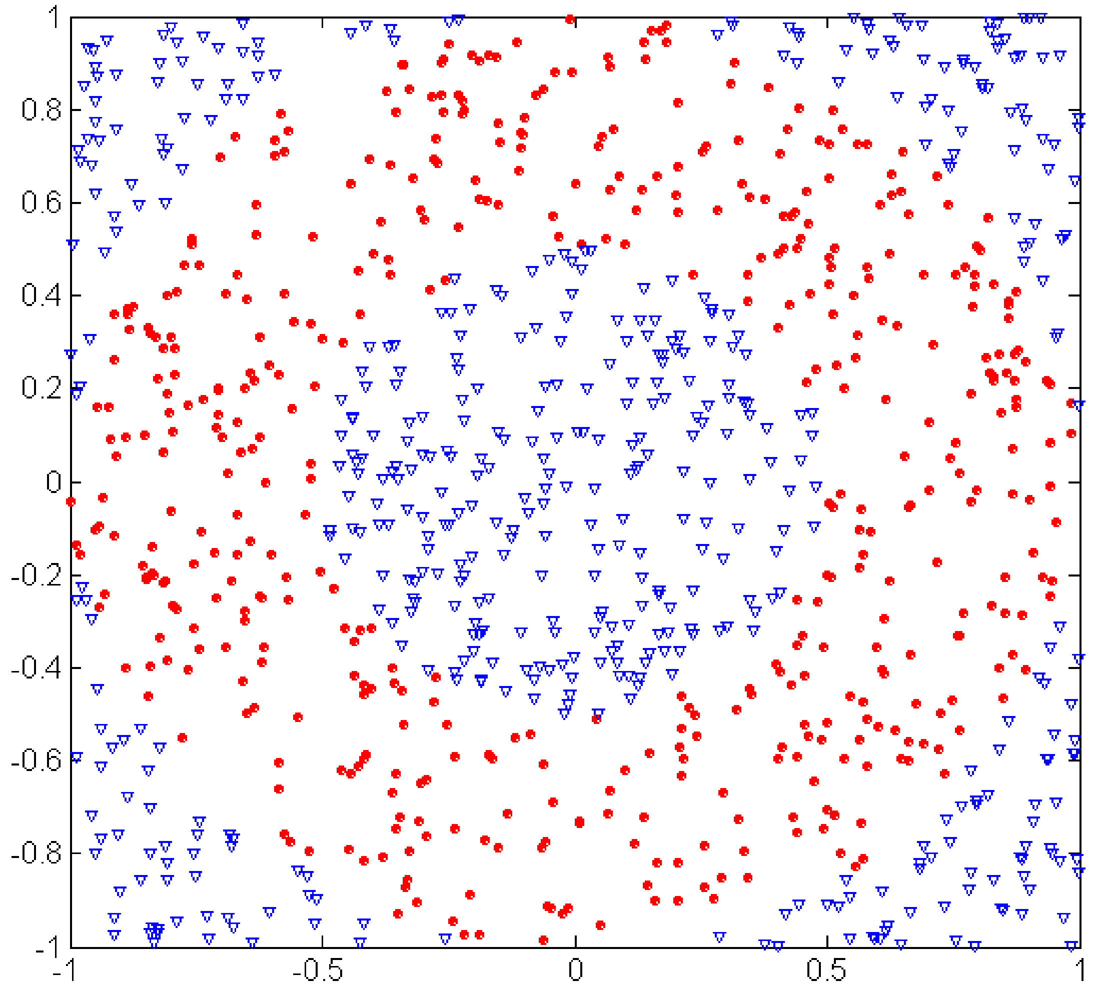
DISADVANTAGES OF DECISION TREE BASED CLASSIFICATION (CART)

- Learners can *create over-complex trees* that do not generalize the data well. This is called *overfitting*.
- Unstable because *small variations in the data might result in a completely different tree being generated.* This is called *variance*, which needs to be *lowered by methods like bagging and* ***boosting***.
- Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.
- Decision tree learners create *biased trees if some classes dominate*. It is therefore recommended to balance the data set prior to fitting with the decision tree.

PRACTICAL ISSUES OF CLASSIFICATION

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

UNDERFITTING AND OVERFITTING (EXAMPLE)



500 circular and 500 triangular data points.

Circular points:

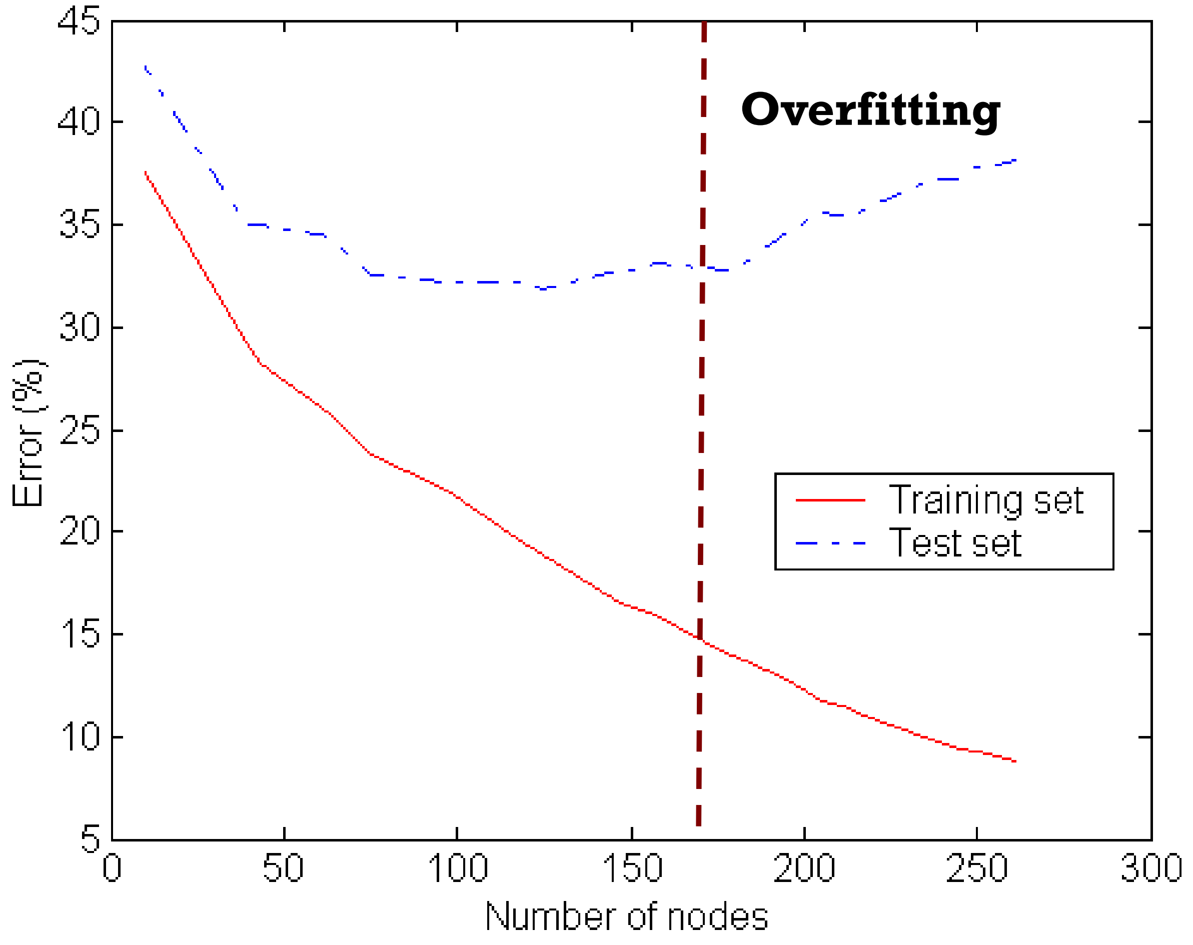
$0.5 \leq \mathrm{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\mathrm{sqrt}(x_1^2+x_2^2) > 0.5$ or

$\mathrm{sqrt}(x_1^2+x_2^2) < 1$

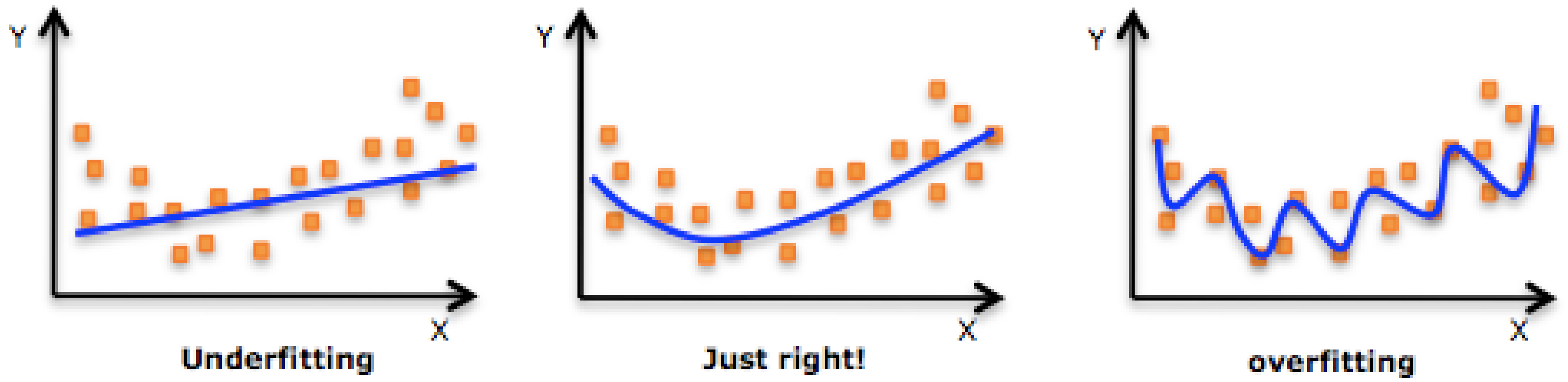
UNDERFITTING AND OVERFITTING



Underfitting: when model is too simple, both training and test errors are large

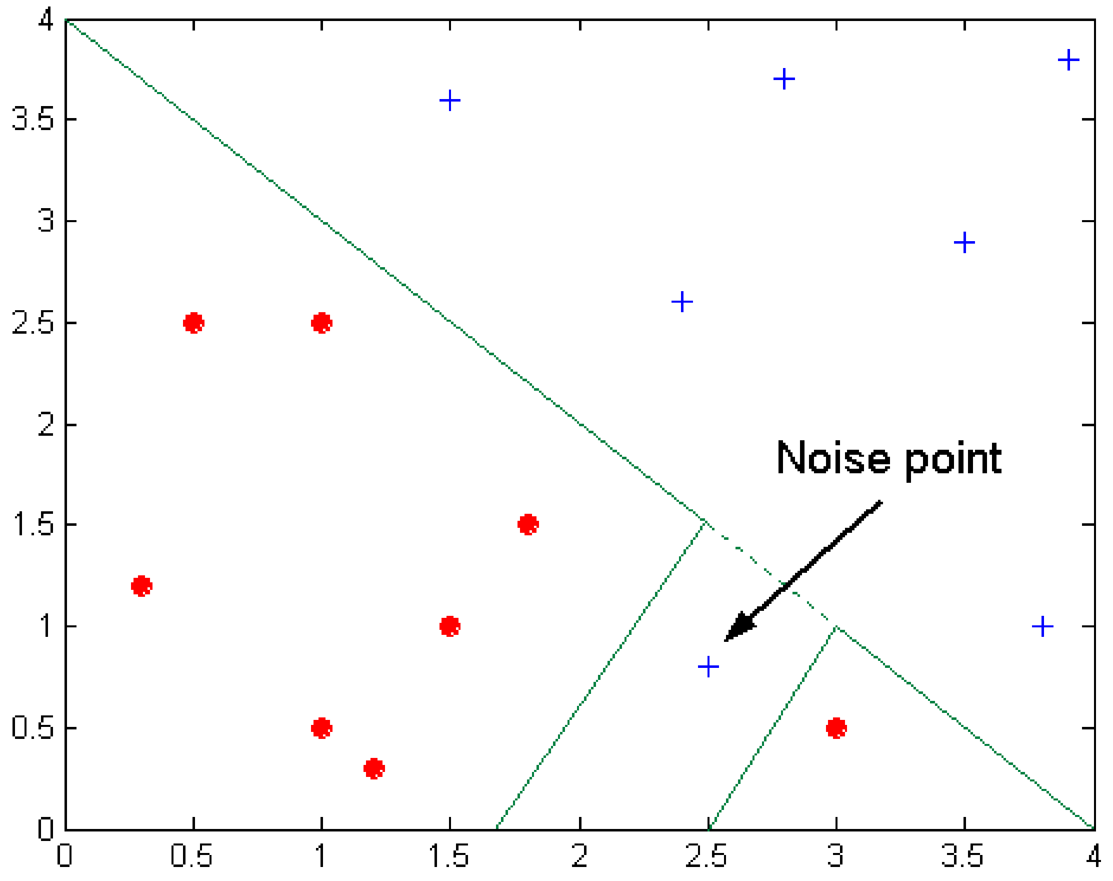
UNDERFITTING AND OVERFITTING

- Learn the “data” and not the underlying function
- Performs well on the data used during the training and poorly with new data.



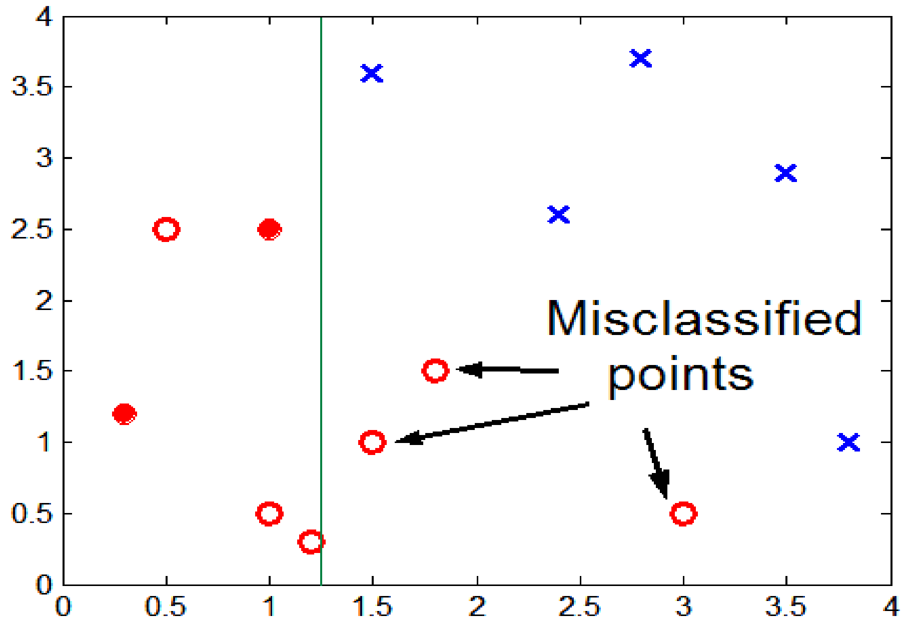
How to avoid it: use proper subsets, early stopping.

OVERFITTING DUE TO NOISE



Decision boundary is distorted by noise point

OVERFITTING DUE TO INSUFFICIENT EXAMPLES



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

NOTES ON OVERFITTING

- Overfitting results in decision trees that are more complex than necessary

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating errors