In statistics and machine learning, one of the most common tasks is to fit a "model" to a set of training data, so as to be able to make reliable predictions on general untrained data.

In overfitting, a statistical model describes random error or noise instead of the underlying relationship.

Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

1

A model that has been overfit has poor predictive performance, as it overreacts to minor fluctuations in the training data.

The possibility of overfitting exists because the criterion used for training the model is not the same as the criterion used to judge the efficacy of a model.

In particular, a model is typically trained by maximizing its performance on some set of training data.

2

However, its efficacy is determined not by its performance on the training data but by its ability to perform well on unseen data.

Overfitting occurs when a model begins to "memorize" training data rather than "learning" to generalize from trend.

3

As an extreme example, if the number of parameters is the same as or greater than the number of observations, a simple model or learning process can perfectly predict the training data simply by memorizing the training data in its entirety, but such a model will typically fail drastically when making predictions about new or unseen data, since the simple model has not learned to generalize at all.

4

The potential for overfitting depends not only on the number of parameters and data but also the conformability of the model structure with the data shape, and the magnitude of model error compared to the expected level of noise or error in the data.

Even when the fitted model does not have an excessive number of parameters, it is to be expected that the fitted relationship will appear to perform less well on a new data set than on the data set used for fitting.

5

In particular, the value of the coefficient of determination will shrink relative to the original training data.

In order to avoid overfitting, it is necessary to use additional techniques (e.g. cross-validation, regularization, early stopping, pruning, Bayesian priors on parameters or model comparison), that can indicate when further training is not resulting in better generalization.

6

The basis of some techniques is either:

(1) to explicitly penalize overly complex models, or

(2) to test the model's ability to generalize by evaluating its performance on a set of data not used for training, which is assumed to approximate the typical unseen data that a model will encounter.

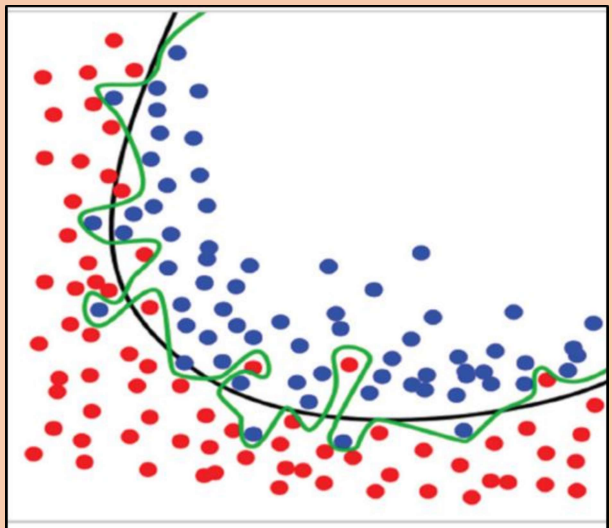
7

A good analogy for the overfitting problem is imagine **a baby trying to learn what is a window or what is not a window**, we start to show him windows and he detects at an initial phase that all windows have glasses, and a frame and you can look outside, some of them may be opened.

If we keep showing the same windows the baby may also falsely deduce that all windows are green, and that all green frames are windows.

Thus overfitting the problem.

8



The green line represents an overfitted model and the black line represents a regularised model. While the green line best follows the training data, it is too dependent on it and it is likely to have a higher error rate on new unseen data, compared to the black line.

9

Overfitting is especially likely in cases where **learning was performed too long** or **where training examples are rare**, **causing the learner to adjust to very specific random features of the training data, that have no causal relation to the target function**.

In this process of overfitting, the **performance on the training examples still increases while the performance on unseen data becomes worse**.

10

As a simple example, consider a database of retail purchases that includes the item bought, the purchaser, and the date and time of purchase.

It's easy to construct a model that will fit the training set perfectly by using the date and time of purchase to predict the other attributes; but this model will not generalize at all to new data, because those past times will never occur again.

Generally, **a learning algorithm is said to overfit relative to a simpler one if it is more accurate in fitting known data (hindsight) but less accurate in predicting new data (foresight)**.

11

One can intuitively understand overfitting from the fact that information from all past experience can be divided into two groups: information that is relevant for the future and irrelevant information ("noise").

Everything else being equal, the more difficult a criterion is to predict (i.e., the higher its uncertainty), the more noise exists in past information that needs to be ignored.

The problem is determining which part to ignore.

12

A learning algorithm that can reduce the chance of fitting noise is called robust.

Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data.

It occurs when the model or algorithm does not fit the data enough.

13

Underfitting occurs if the model or algorithm shows low variance but high bias (to contrast the opposite, overfitting from high variance and low bias).

It is often a result of an excessively simple model.

14

## Overfitting

Consider error of hypothesis $h$ over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

15

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize
  $size(tree) + size(misclassifications(tree))$

16

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

17

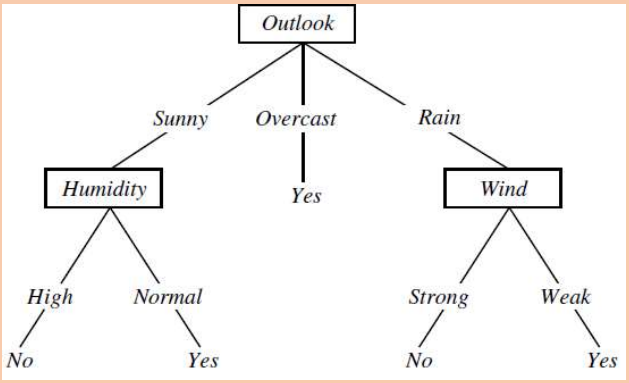
**Rule: Post Pruning**

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

18

Converting a tree to rules




IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

19

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge, \vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- $M$ of $N$

20

## When to consider decision Trees?

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

21

## Classification Trees

In classification tree the case of a decision tree for classification, namely, a ***classification*** **impurity measure *tree***, **the goodness of a split is quantified by an *impurity measure***.

A split is pure if after the split, for all branches, all the instances choosing a branch belong to the same class.

22

It can also be said that at each step during tree construction, we choose the split that causes the largest decrease in impurity, which is the difference between the impurity of data reaching node *m* and the total entropy of data reaching its branches after the split.

One problem is that such splitting favors attributes with many values.

When there are many values, there are many branches, and the impurity can be much less.

23

Nodes with many branches are complex and go against our idea of splitting class discriminants into simple decisions.

Methods have been proposed to penalize such attributes and to balance the impurity drop and the branching factor.

24

## Pruning

Frequently, a node is not split further if the number of training instances reaching a node is smaller than a certain percentage of the training set - for example, 5 percent—regardless of the impurity or error.

The idea is that any decision based on too few instances causes variance and thus generalization error.

Stopping tree construction early on before it is full is called ***prepruning*** the tree.

25

Another possibility to get simpler trees is *postpruning*, which in practice works better than prepruning.

Tree growing is greedy and at each step, we make a decision, namely, generate a decision node, and continue further on, never backtracking and trying out an alternative.

The only exception is postpruning where we try to find and prune unnecessary subtrees.

26

In postpruning, we grow the tree full until all leaves are pure and we have no training error. We then find subtrees that cause overfitting and we prune them.

From the initial labeled set, we set aside a *pruning set*, unused during training.

For each subtree, we replace it with a leaf node labeled with the training instances covered by the subtree (appropriately for classification or regression).

27

If the leaf node does not perform worse than the subtree on the pruning set, we prune the subtree and keep the leaf node because the additional complexity of the subtree is not justified; otherwise, we keep the subtree.

Comparing prepruning and postpruning, we can say that prepruning is faster but postpruning generally leads to more accurate trees.

28

## Rule Extraction from Trees

A decision tree does its own feature extraction.

The univariate tree only uses the necessary variables, and after the tree is built, certain features may not be used at all.

We can also say that features closer to the root are more important globally.

For example, the decision tree given in figure below uses $x1$, $x2$, and $x4$, but not $x3$.

29

It is possible to use a decision tree for feature extraction: we build a tree and then take only those features used by the tree as inputs to another learning method.

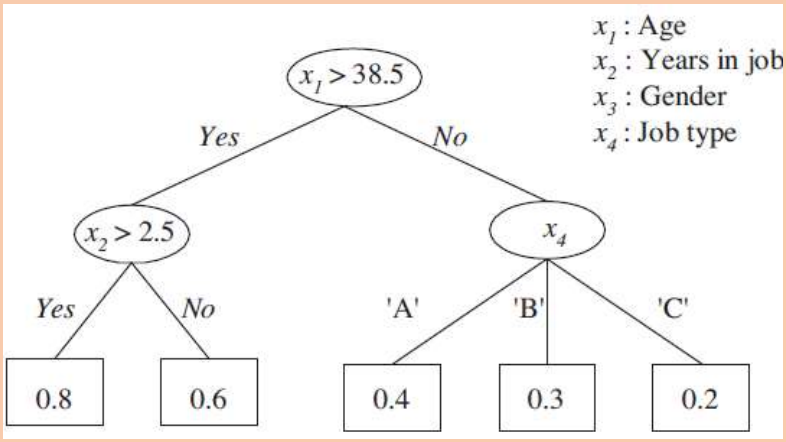



**Figure:** Example of a (hypothetical) decision tree. Each path from the root to a leaf can be written down as a conjunctive rule, composed of conditions defined by the decision nodes on the path.

30

Another main advantage of decision trees is *interpretability*: The decision nodes carry conditions that are simple to understand.

Each path from the root to a leaf corresponds to one conjunction of tests, as all those conditions should be satisfied to reach to the leaf.

These paths together can be written down as a set of *IF-THEN rules*, called a *rule base*.

One such method is *C4.5Rules* (Quinlan 1993).

31

32

33

34

35

36

37

38

39

40

41

42

43