Underfitting and Overfitting (Example)

500 circular and 500 triangular data points.

Circular points:
\[ 0.5 \leq \sqrt{x_1^2 + x_2^2} \leq 1 \]

Triangular points:
\[ \sqrt{x_1^2 + x_2^2} < 0.5 \text{ or } \sqrt{x_1^2 + x_2^2} > 1 \]

Underfitting and Overfitting

Overfitting

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

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

Estimating Generalization Errors

- Re-substitution errors: error on training (\( \Sigma e(t) \))
- Generalization errors: error on testing (\( \Sigma e'(t) \))
- Methods for estimating generalization errors:
  - Optimistic approach: \( e'(t) = e(t) \)
    - Pessimistic approach:
      - For each leaf node: \( e'(t) = (e(t)) + 0.5 \)
      - Total errors: \( e'(T) = e(T) + N \times 0.5 \) (\( N \): number of leaf nodes)
      - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
        - Training error = 10/1000 = 1%
        - Generalization error = (10 + 30 \times 0.5)/1000 = 2.5%
  - Reduced error pruning (REP):
    - uses validation data set to estimate generalization error
Occam’s Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.
- For complex models, there is a greater chance that it was fitted accidentally by errors in data.
- Therefore, one should include model complexity when evaluating a model.

How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree.
  - Possible conditions:
    - Stop if number of instances is less than some user-specified threshold.
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain) by at least some threshold.

Disadvantage of Pre-Pruning

- Since we use a hill-climbing search, looking only one step ahead, pre-pruning might stop too early.
- Extreme example: exclusive OR, A XOR B.

How to Address Overfitting...

- Post-pruning
  - Grow decision tree to its entirety.
  - Trim the nodes of the decision tree in a bottom-up fashion.
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree.

Example of Post-Pruning

- Optimistic error?
- Pessimistic error?
- Reduced error pruning?

Examples of Post-pruning

- Case 1:
  - Optimistic error?
  - Pessimistic error?
  - Reduced error pruning?
- Case 2:
  - Optimistic error?
  - Pessimistic error?
  - Reduced error pruning?
Error based pruning in C4.5 (J48)

Suppose we observe 2 errors out of 7 in a leaf node. The point estimate for the error rate in that leaf is 2/7 = 0.286.

If the probability of error is 2/7 then the probability of observing 2 errors or less is 0.68 (binomial distribution with n=7, and p=2/7).

As a pessimistic estimate of the error rate in this leaf node, we are going to find the value of p, such that the probability of 2 errors or less is relatively small, say 0.25. This turns out to be the case for p = 0.4861. Hence [0, 0.4861) can be regarded as a 75% right-sided confidence interval for p.

For p=0.659 the probability of observing 2 errors or less is 0.05. Values higher than 0.659 give even smaller probabilities to observing 2 errors or less, and are therefore highly unlikely.

Handling Missing Attribute Values

• Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

Computing Impurity Measure

Before Splitting:
Entropy(Refund=Yes) = -0.5 log(0.3)-(0.7)log(0.7) = 0.8813

Split on Refund:
Entropy(Refunds=Yes) = -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183
Entropy(Children) = 0.3(0) + 0.6(0.9183) = 0.551
Gain = 0.9 x (0.8813 – 0.551) = 0.3303
**Distribute Instances**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Probability that Refund=Yes is 3/9
Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

**Classify Instances**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

New record:

- Probability that Marital Status = Married is 3/6.67
- Probability that Marital Status = {Single, Divorced} is 3.67/6.67

**Decision Boundary**

- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

For the final prediction for case 11, take the weighted average of the predictions in node A and B.

\[ P_A(\text{class=yes}) = 1, P_B(\text{class = yes}) = 0. \]

Weighted average:

\[ P(\text{class=yes}) = \frac{0.55 \cdot 2.67 + 1 \cdot 0.45 + 3 \cdot 0}{0.55 \cdot 2.67 + 0.45 + 3} \approx 0.52 \]
Oblique Decision Trees

- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

\[
\begin{array}{c|cc}
\text{ACTUAL CLASS} & \text{Class=Yes} & \text{Class=No} \\
\hline
\text{Class=Yes} & a & b \\
\text{Class=No} & c & d \\
\end{array}
\]

- Most widely-used metric:

\[
\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}
\]

Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example
## Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>C(Yes</td>
</tr>
<tr>
<td>Class=No</td>
<td>C(No</td>
</tr>
</tbody>
</table>

C(i|j): Cost of misclassifying class j example as class i

## Computing Cost of Classification

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 150 40</td>
<td>- 60 250</td>
</tr>
</tbody>
</table>

Model M1: PREDICTED CLASS

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 250 45</td>
<td>- 5 200</td>
</tr>
</tbody>
</table>

Model M2: PREDICTED CLASS

Accuracy = 80%
Cost = 3910

Accuracy = 90%
Cost = 4255

## Cost-Sensitive Measures

**True positive rate**: TP/(TP+FN), Fraction of positive examples that is predicted to be positive.

**False positive rate**: FP/(FP+TN), Fraction of negative examples that is predicted to be positive.

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>a (TP) b (FN)</td>
</tr>
<tr>
<td>Class=No</td>
<td>c (FP) d (TN)</td>
</tr>
</tbody>
</table>

## ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
- Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
- Changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

### ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- Any points located at x > 1 is classified as positive

At threshold t:
TP=0.5, FN=0.5, FP=0.12, TN=0.88

### ROC Curve

(TP,FP):
- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line: Prediction is opposite of the true class
Using ROC for Model Comparison

- No model consistently outperform the other
- \( M_1 \) is better for small FPR
- \( M_2 \) is better for large FPR
- Area Under the ROC curve
  - Ideal: \[ \text{Area} = 1 \]
  - Random guess: \[ \text{Area} = 0.5 \]

How to Construct an ROC curve

1. Use classifier that produces posterior probability for each test instance \( P(+|A) \)
2. Sort the instances according to \( P(+|A) \) in decreasing order
3. Apply threshold at each unique value of \( P(+|A) \)
4. Count the number of TP, FP, TN, FN at each threshold
5. TP rate, TPR = \( \frac{TP}{TP + FN} \)
6. FP rate, FPR = \( \frac{FP}{FP + TN} \)

| Instance | \( P(+|A) \) | True Class | FPR | TPR |
|----------|---------------|------------|-----|-----|
| 1        | 0.95          | +          | 0   | 1/5 |
| 2        | 0.93          | +          | 0   | 2/5 |
| 3        | 0.87          | +          | 1/5 | 2/5 |
| 4        | 0.85          | -          |     |     |
| 5        | 0.85          | -          |     |     |
| 6        | 0.85          | +          | 3/5 | 3/5 |
| 7        | 0.76          | -          | 4/5 | 3/5 |
| 8        | 0.53          | +          | 4/5 | 4/5 |
| 9        | 0.43          | -          | 1   | 4/5 |
| 10       | 0.25          | +          | 1   | 1   |