Metrics for model evaluation

Starting from the basic logic to completely define an algorithm for building decision trees it is necessary to define:

- The split condition
- The criterion defining the best split
- The criterion for stopping the split
- **Methods for evaluating the goodness of a decision tree**

Metrics for model evaluation

The Confusion Matrix evaluates the ability of a classifier based on the following indicators

- TP (true positive): records correctly classified as Yes class
- FN (false negative): Incorrectly classified records as class No
- FP (false positive): Incorrectly classified records as class Yes
- TN (true negative) records correctly classified as class No

		Expected Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	TP	FN
	Class=No	FP	TN

If the classification uses n classes, the confusion matrix will be $n \times n$

Accuracy

Accuracy is the most widely used metric to synthesize the information of a confusion matrix

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Equally, the frequency of the error could be used

$$\text{Error rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

Accuracy Limitiations

Accuracy is not an appropriate metric if the classes contain a very different number of records. Consider a binary classification problem in which

- # Record of class 0 = 9990
- # Record of class 1 = 10

A model that always returns class 0 will have an accuracy of 9990/10000 = 99.9%

In the case of binary classification problems, the class "rare" is also called a positive class, while the class that includes most of the records is called a negative class

Precision and Recall

Precision and Recall are two metrics used in applications where the correct classification of positive class records is more important

Precision measures the fraction of record results actually positive among all those who were classified as such

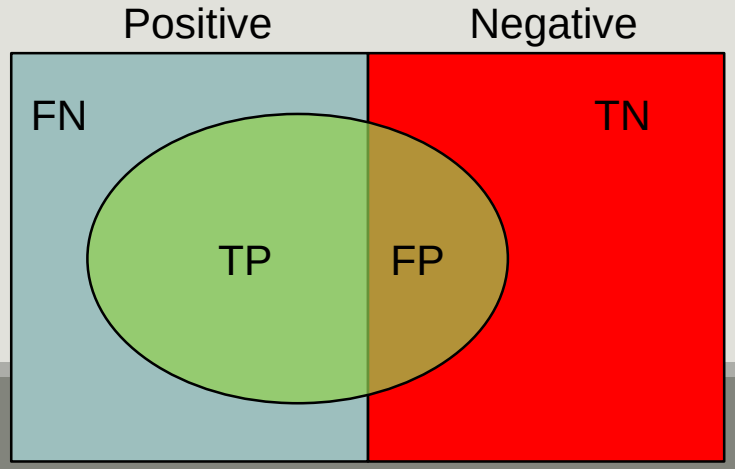
- High values indicate that few negative class records were incorrectly classified as positive.

Recall measures the fraction of positive records correctly classified

- High values indicate that few records of the positive class were incorrectly classified as negatives.

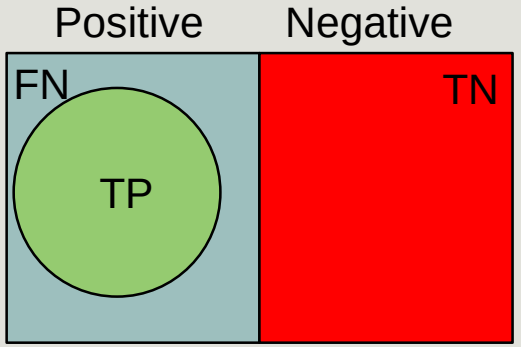
$$\text{Precision, } p = \frac{TP}{TP + FP}$$

$$\text{Recall, } r = \frac{TP}{TP + FN}$$

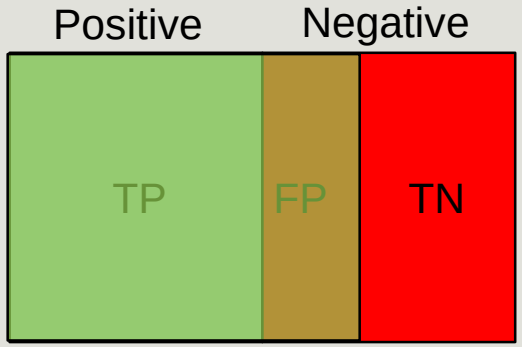


Precision and Recall

Precision = 1 if all the positive records were actually detected



Recall = 1 if there are no false negatives



If both are valid 1, the predetermined classes coincide with the real ones

F-measure

A metric that summarizes precision and recall is called F-measure

$$\text{F-measure, F} = \frac{2rp}{r+p} = \frac{2\times TP}{2\times TP + FP + FN}$$

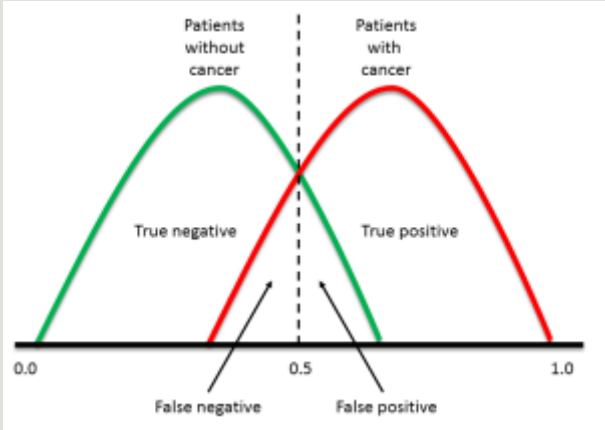
F-measure represents the harmonic mean of precision and recall

- The harmonic average between two x and y numbers tends to be close to the smallest of the two numbers. So if the harmonic average is high, it means both precision and recall are.
  - ... so there have been no false negative or false positives

Cost-Based Evaluation

Accuracy, Precision-Recall and F-measure classify an instance as positive if P(+,i)>P(-,i).

- They assume that FN and FP have the same weight, thus they are Cost-Insensitive
- In many domains this is not true!
  - For a cancer screening test, for example, we may be prepared to put up with a relatively high false positive rate in order to get a high true positive, it is most important to identify possible cancer sufferers
  - For a follow-up test after treatment, however, a different threshold might be more desirable, since we want to minimize false negatives, we don't want to tell a patient they're clear if this is not actually the case.



The Cost Matrix

The cost matrix encodes the penalty that a classifier incurs in classifying a record in a different class
A negative penalty indicates the "prize" that is obtained for a correct classification

$$C(M)=TP\times C(+|+) + FP\times C(+|-) + FN\times C(-|+) + TN\times C(-|-)$$

	Expected Class j		
Actual Class i	C(i|j)	**Class = +**	**Class=-**
	Class=+	C(+|+)	C(+|-)
	Class=-	C(-|+)	C(-|-)

A model constructed by structuring, as a purity function, a cost matrix will tend to provide a model with a minimum cost over the specified weights

Computing the Cost

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Accuracy = 80%

Cost = 3910

Model M_1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 90%

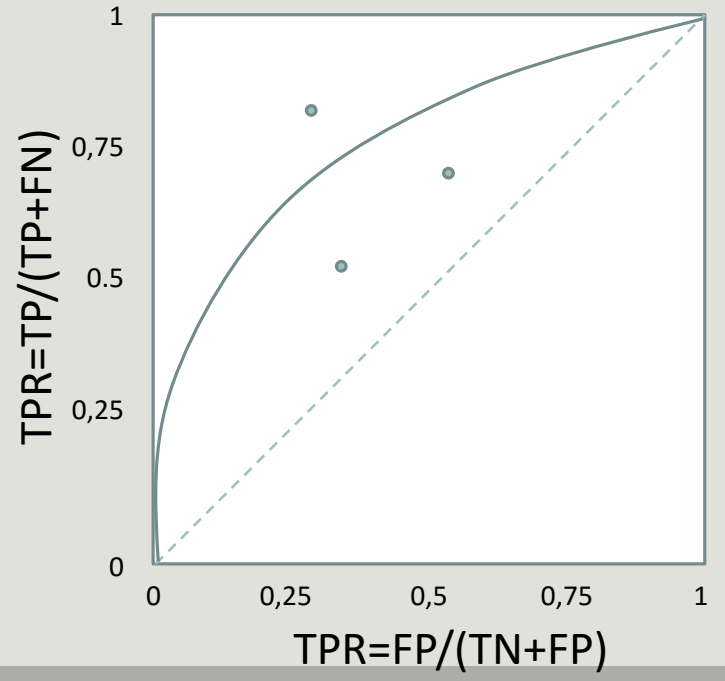
Cost = 4255

ROC Space

ROC graphs are two-dimensional graphs that depict relative tradeoffs between benefits (true positive) and costs (false positive) induced by a classifier. We distinguish between:

- ***Probabilistic classifiers*** return a score that is not necessarily a *sensu stricto* probability but represents the degree to which an object is a member of one particular class rather than another one (e.g. Decision tree, Naïve Bayes)
  - In a decision tree an istance in a leaf is associated to the class + if the number positive training instances in the leaf (*pos*) is greather than the number of negative instances (*neg*). The ratio *pos*/(*pos*+*neg*) can be used as a score showing the likelihood of an instance to be of class + or -
- A ***discrete classifier*** predicts only the classes to which a test object belongs (e.g. SVM)

ROC curve characterizes a probabilistic classifier, and each point of this curve corresponds to a discrete classifier.



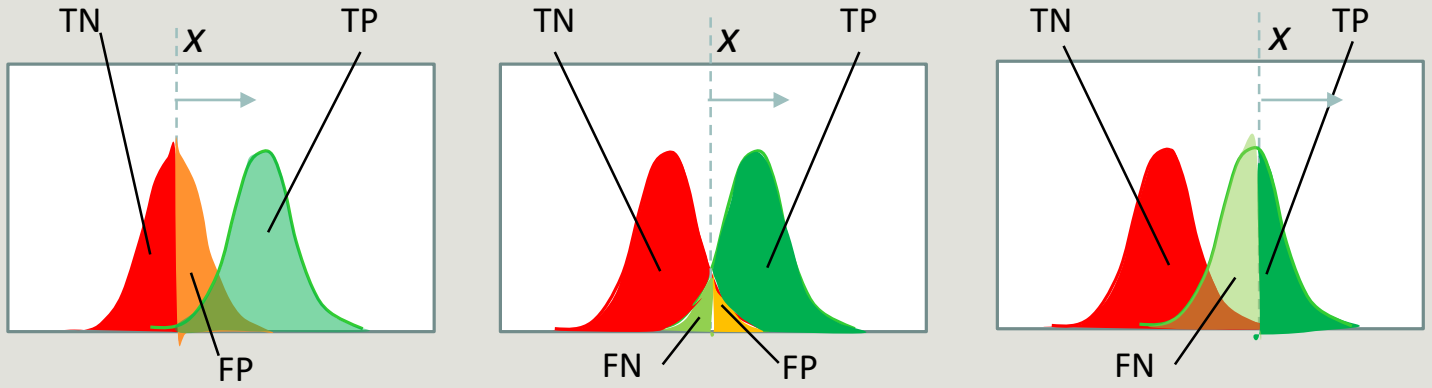
ROC Space

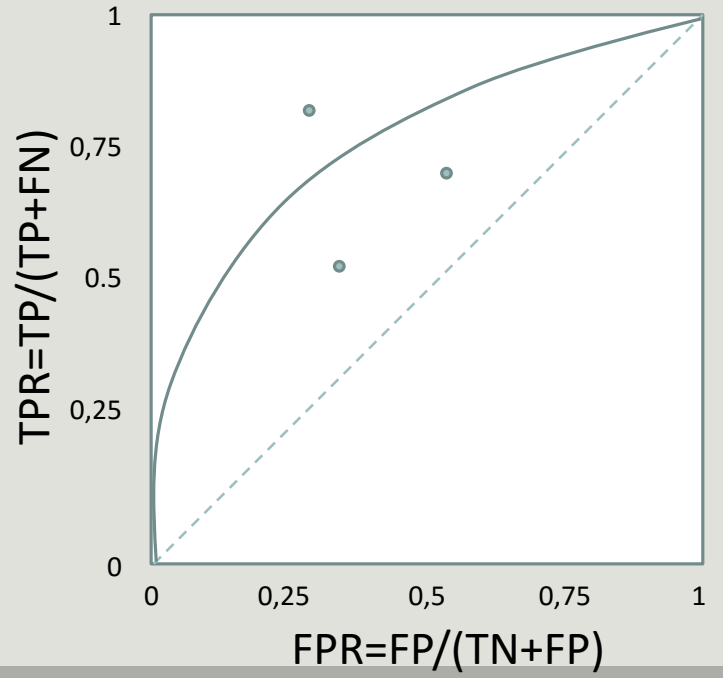
A ROC graph for a probabilistic classifier is obtained varying the threshold (or the probability if available) used to assign an instance *i* to a class (+/-).

- Instead of P(+,*i*) > P(-,*i*) than *i* is +
- We have if P(+,i)> *x* than *i* is + $x \in [0,..,1]$

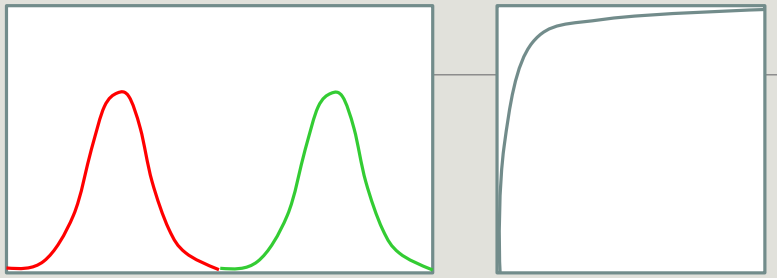
Each *x* value determines different TPR and FPR

The ROC curve shape depends both on the classifier capabilities and on the dataset features.

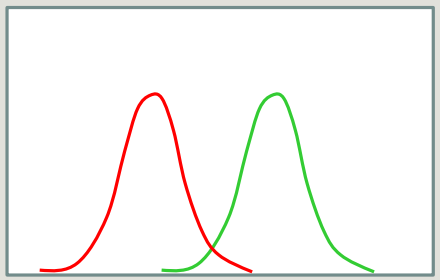


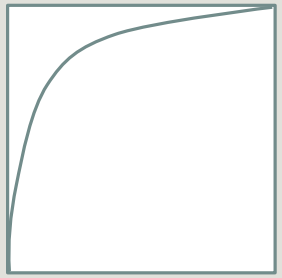


Understanding the ROC Space

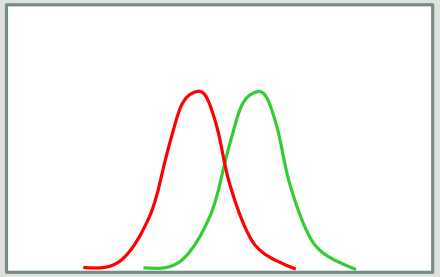


A good classifier tends to have performance close to the higher-left corner of the ROC graph that is: High TPR and Low FPR

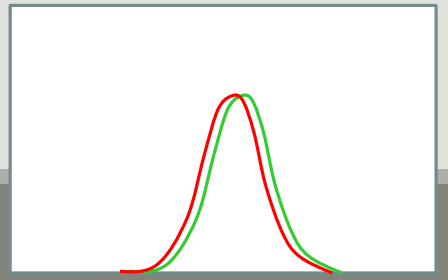


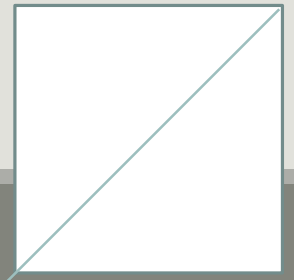


The **Area Under the Curve** – AUC provides an overall rating of the classifier, while segments of the curve provide a rating in specific TPR – FPR settings.



The larger the overlap between + and – instances distributions, the harder for the classifier to distinguish between positive and negative instances.



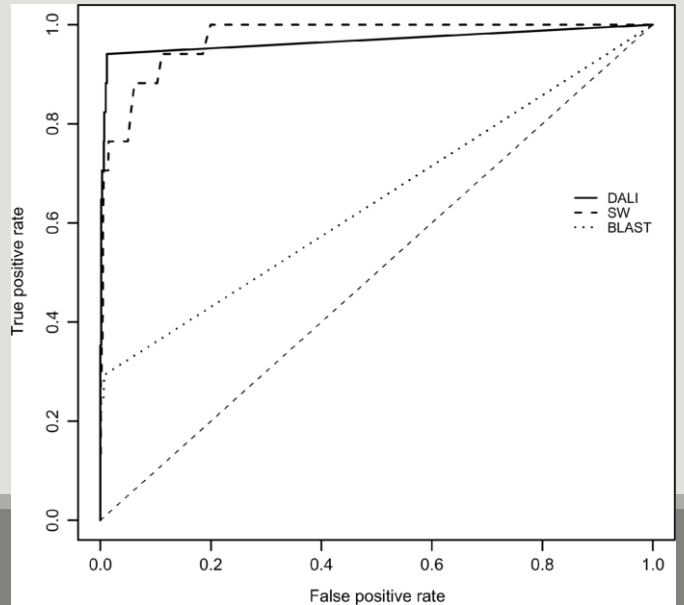


A dummy classifier performs on the ROC graph diagonal: the TPR is equal to the FPR since the classifier answers are random

Comparison of Classifier via ROC curve

A classifier comparison based on ROC curves or AUC values can be either graphical or numerical.

- A ROC curve running above another is an indicator of better classifier performance, and by the same token, the bigger the AUC, the better the overall performance of the test.
- However, this reasoning is meaningful only if the two ROC curves do not cross at any point. If they do, then it makes intuitive sense to point out the region in which one classifier outperforms the other, but the comparison of the complete AUC values is not very informative.
  - DALI is better than SW when a low FP rate is needed
  - BLAST is always worse than DALI & SW



ROC space properties

ROC curves are insensitive to changes in class distribution. If the proportion of positive to negative instances changes in a test set, the ROC curves will not change.

The class distribution is the relationship of the left (P) column to the right (N) column. Any performance metric that uses values from **both columns** will be inherently sensitive to class skews. Metrics such as accuracy, precision and F score use values from both columns of the confusion matrix. As a class distribution changes these measures will change as well, even if the fundamental classifier performance does not.

		True class: p	True class: n	
Hypothesized class	Y	True Positives	False Positives	fp rate $= \frac{FP}{N}$ tp rate $= \frac{TP}{P}$
	N	False Negatives	True Negatives	precision $= \frac{TP}{TP+FP}$ recall $= \frac{TP}{P}$
				accuracy $= \frac{TP+TN}{P+N}$
Column totals:		P	N	F-measure $= \frac{2}{1/\text{precision}+1/\text{recall}}$

ROC graphs are based upon TPR and FPR, in which each dimension is a strict columnar ratio, so do not depend on class distributions.

Where do ROC curves come from?

ROC stands for *Receiver Operator Characteristic*. The term has its roots in World War II. ROC curves were originally developed by the British as part of the "Chain Home" radar system. ROC analysis was used to analyze radar data to differentiate between enemy aircraft and signal noise (e.g. flocks of geese).

Radar Operators were human classifiers!

Classification Errors

Training error: are mistakes that are made on the training set

Generalization error: errors are made on the test set (i.e. records that have not been trained on the system).

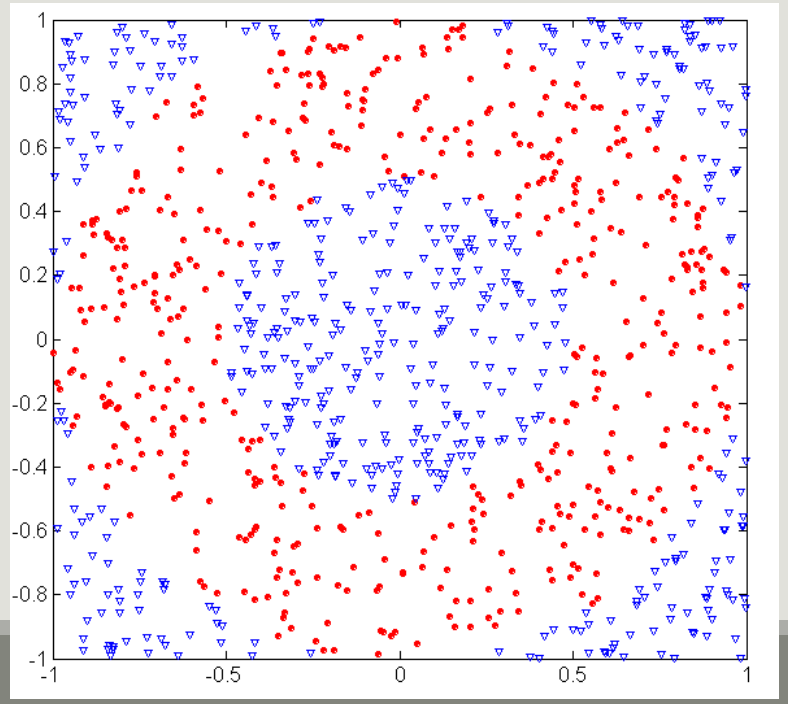
Underfitting: The model is too simple and does not allow a good classification or set training set or test set

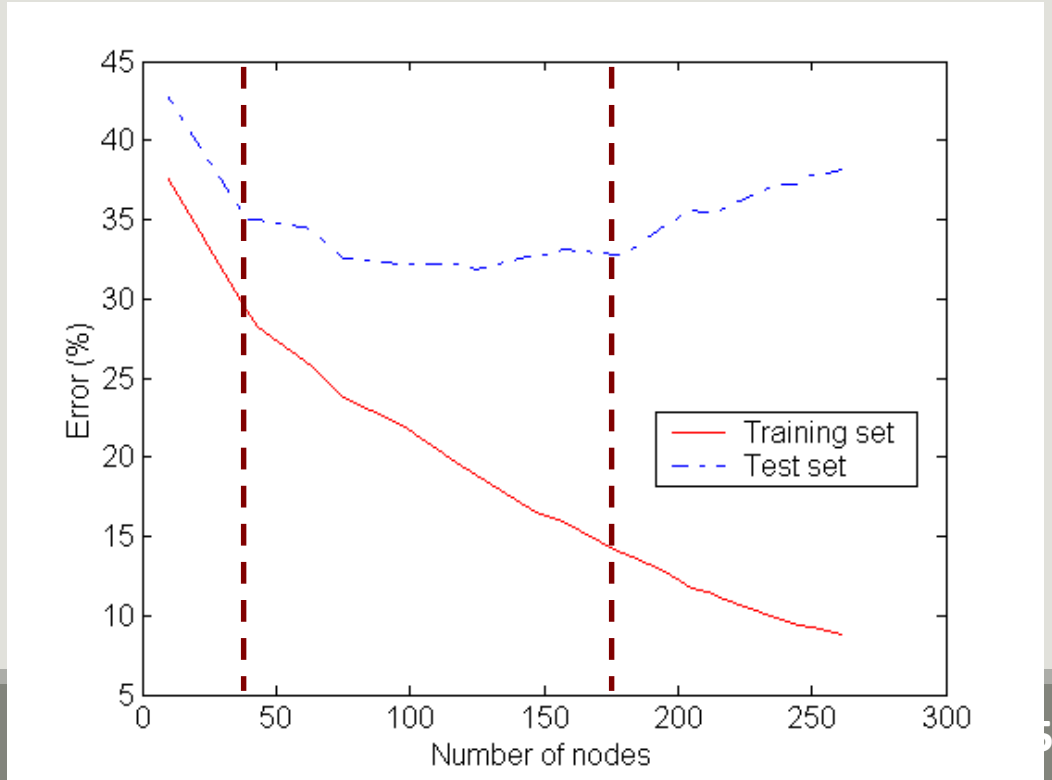
Overfitting: The model is too complex, it allows a good classification of the training set, but a poor classification of the test set

- The model fails to generalize because it is based on the specific peculiarities of the training set that are not found in the test set (e.g. noise present in the training set)

Underfitting and Overfitting

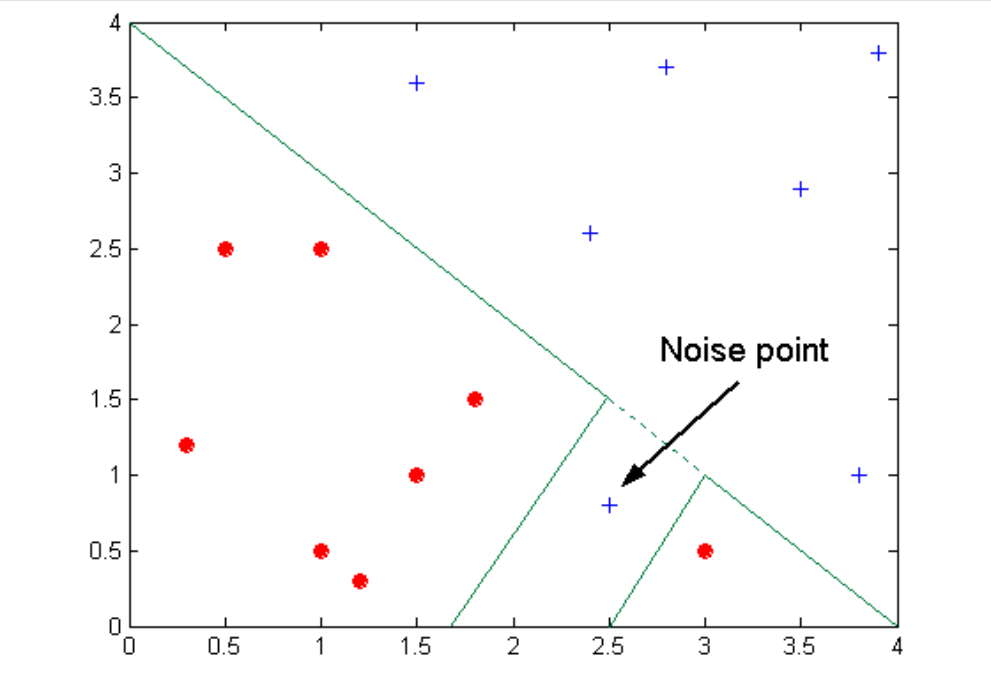
- 500 circles and 500 triangles
- Circular points: $0.5 \leq \text{sqrt}(x12+x22) \leq 1$
- Triangular points: $\text{sqrt}(x12+x22) > 0.5$ or $\text{sqrt}(x12+x22) < 1$





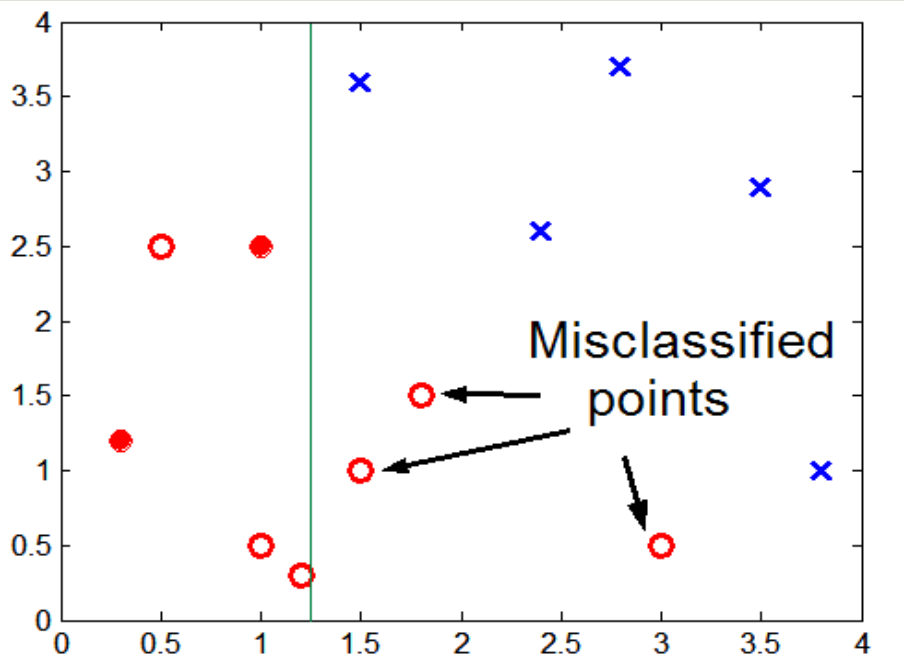
Overfitting Due to Noise

The boundaries of the areas are distorted due to noise



Overfitting due to the reduced size of the training set

Lack of points at the bottom of the chart makes it difficult to find a proper classification for that portion of the region



Points in the training set

How to handle the Overfitting: pre-pruning (Early stopping rule)

Stop splitting before you reach a deep tree

A node can not be split further if:

- Node does not contain instances
- All instances belong to the same class
- All attributes have the same values

More restrictive conditions potentially applicable are:

- Stop splitting if the number of instances in the node is less than a fixed amount
- Stop splitting if distribution of instances between classes is independent of attribute values
- Stop splitting if you do not improve the purity measure (e.g. Gini or information gain).

How to handle the Overfitting: post-pruning (Reduced Error Pruning)

Run all possible splits
Examine the decision tree nodes obtained with a bottom-up logic
Collate a sub tree in a leaf node if this allows to reduce the generalization error (i.e. on the validation set)

- Choose to collapse the sub tree that determines the maximum error reduction (N.B. greedy choice)

Instances in the new leaf can be tagged

- Based on the label that appears most frequently in the sub-tree
- According to the label that occurs most frequently in the instances of the training set that belong to the sub-tree

Post-pruning is more effective but involves more computational cost. It is based on the evidence of the result of a complete tree

Notes on Overfitting

Overfitting results in more complex decision-making trees than necessary

The classification error done on the training set does not provide accurate estimates about tree behavior on unknown records

It requires new techniques to estimate generalization errors

Estimate generalization errors

A decision tree should minimize the error on the real data set, unfortunately during construction, only the training set is available.
Then the real time data error must be estimated.

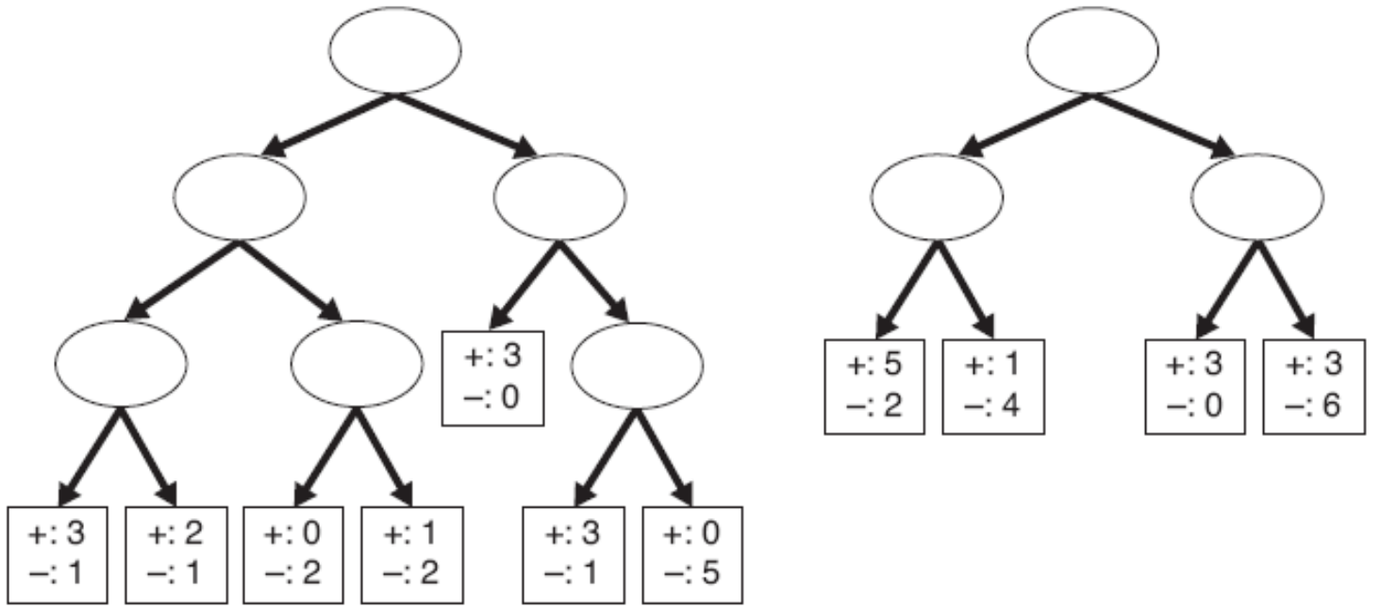
- **Re-substitution error:** number of errors in the training set
- **Generalization error:** number of errors in the real data set

The methods for estimating the generalization error are:

- **Optimistic approach:** e'(t) = e(t)
- **Pessimistic approach**
- **Minimum Description Length (MDL)**
- **Using the test set:** The generalization error is equal to the error on the test set.
  - Normally the test set is obtained by extracting from the initial training set 1/3 of the records
  - It offers good results but the risk is to work with a too small training set

Generalization Error: Resubstitution Estimate

- Just use the training error



Decision Tree, T_L Decision Tree, T_R

Example of two decision trees generated from the same training data.

$$e(T_L) = 4/24 = 0.167$$

$$e(T_R) = 6/24 = 0.25$$

Occam's Razor

- Given two models of similar generalization errors one should prefer the simpler model over the more complex model

Ockham chooses a razor

- 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

Generalization Error: Pessimistic Estimate

- For each leaf node: e'(t) = (e(t)+0.5)
- Total errors: e'(T) = e(T) + N × 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 + 30×0.5)/1000 = 2.5%

Add a "penalty" to no. of misclassified records at each leaf node

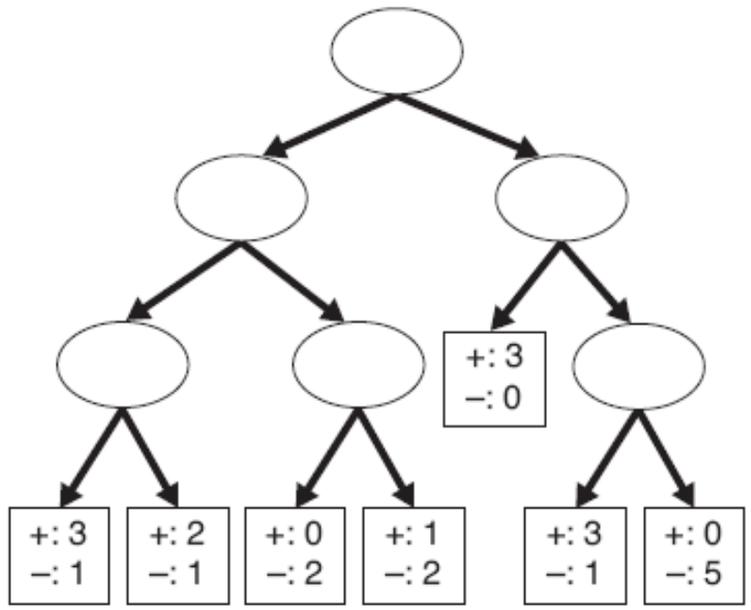
Generalization Error: Pessimistic Estimate

- For each leaf node: e'(t) = (e(t)+0.5)
- Total errors: e'(T) = e(T) + N × 0.5 (N: number of leaf nodes)
- For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):

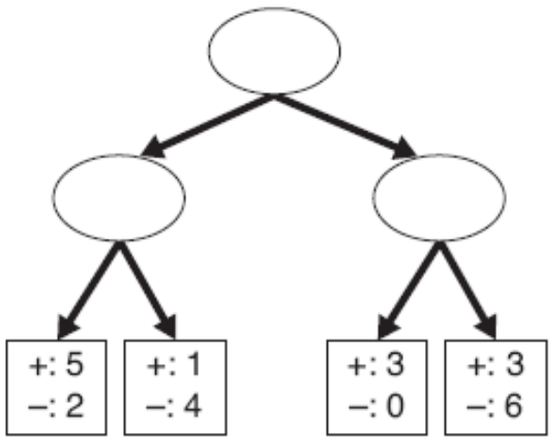
  Training error = 10/1000 = 1%

  Generalization error = (10 + 30×0.5)/1000 = 2.5%

Add a "penalty" to no. of misclassified records at each leaf node



Decision Tree, T_L



Decision Tree, T_R

$$e_p(T_L) = \frac{4 + 7 \times 0.5}{24} = 0.3125$$

$$e_p(T_R) = \frac{6 + 4 \times 0.5}{24} = 0.3333$$

Example of two decision trees generated from the same training data.

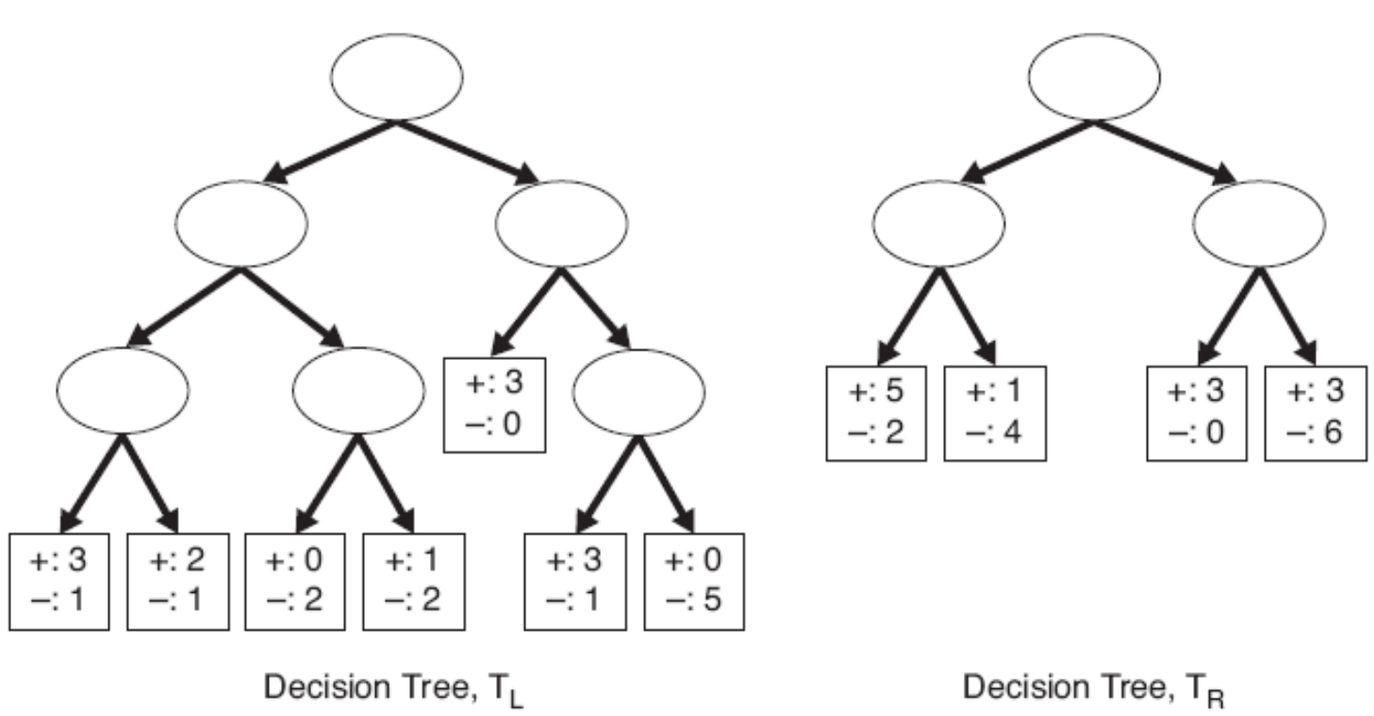
Generalization Error: Pessimistic Estimate

- For each leaf node: e'(t) = (e(t)+0.5)
- Total errors: e'(T) = e(T) + N × 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 + 30×0.5)/1000 = 2.5%

Can be some other number e.g. 1



Example of two decision trees generated from the same training data.

$$e_p(T_L) = \frac{4 + 7 \times 1}{24} = 0.458$$

$$e_p(T_R) = \frac{6 + 4 \times 1}{24} = 0.417$$

Generalization Error: Validation Set

- Instead of using training set to compute generalization error (resubstitution estimate), split the training data into two components
  - one for model-building, and one, a “validation set,” for computing generalization error
  - choose one among a set of model choices (e.g. different sized trees) by comparing their errors on the validation set
  - Caveat: less training data available for model building

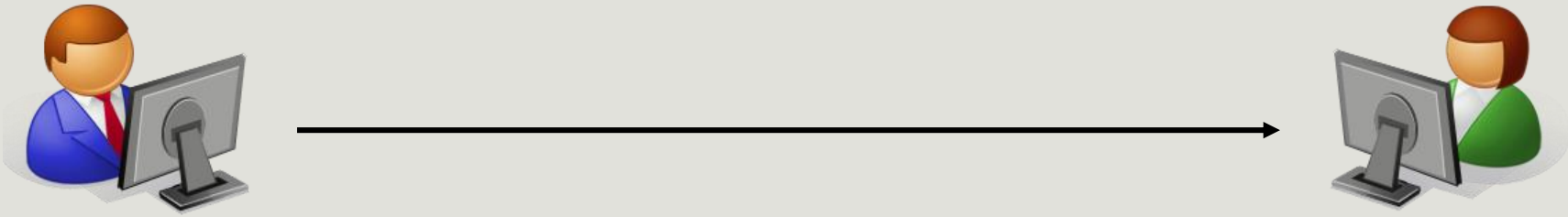
Minimum Description Length

Give two models choose the one that minimizes the cost to describe a classification
To describe the model I can:

A) Sequentially send class O (n)

B) Build a classifier, and send the description along with a detailed description of the mistakes it makes

Cost(*model*,*data*)=Cost(*model*)+Cost(*data*|*model*)



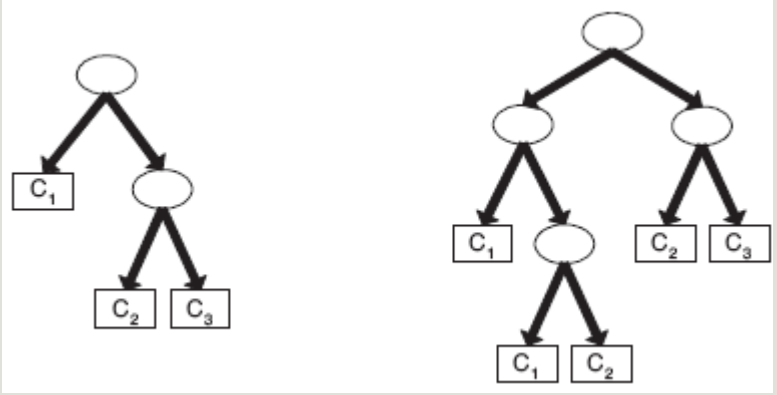
X	y
X1	1
X2	0
X3	1
…	…
Xn	0

X	y
X1	?
X2	?
X3	?
…	…
Xn	?

Minimum Description Length

Datasets with n records described by 16 binary attributes and 3 class values

Tree1
7 errors



Tree2
4 errors

- Each inner node is modeled with the ID of the used attribute → log2(16)=4 bit
- Each leaf is modeled with the ID of the class→ log2(3)=2 bit
- Each error is modeled with its position in the training set considering n record→ log2(n)

$$\text{Cost(Tree1)} = 4\times2 + 2\times3 + 7 \times \log2(n) = 14 + 7 \times \log2(n)$$

$$\text{Cost(Tree2)} = 4\times4 + 2\times5 + 4 \times \log2(n) = 26 + 4 \times \log2(n)$$

$$\text{Cost(Tree1)} < \text{Cost(Tree2)} \quad \text{se } n < 16$$

How to Address Overfitting I

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree

How to Address Overfitting I

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same

How to Address Overfitting I

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting II

- 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
  - Can use MDL for post-pruning

Example: Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30

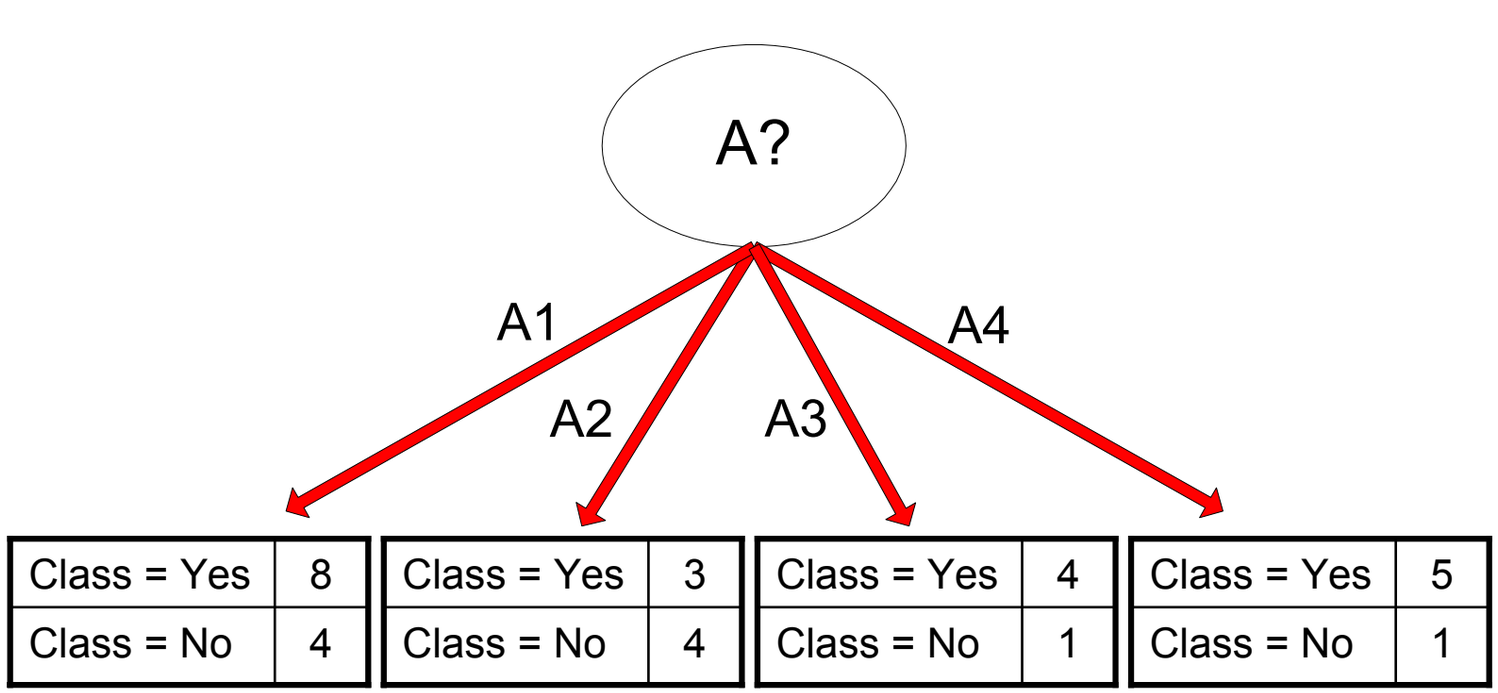
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Example: Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30

Pessimistic error = (10 + 0.5)/30 = 10.5/30



Example: Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30

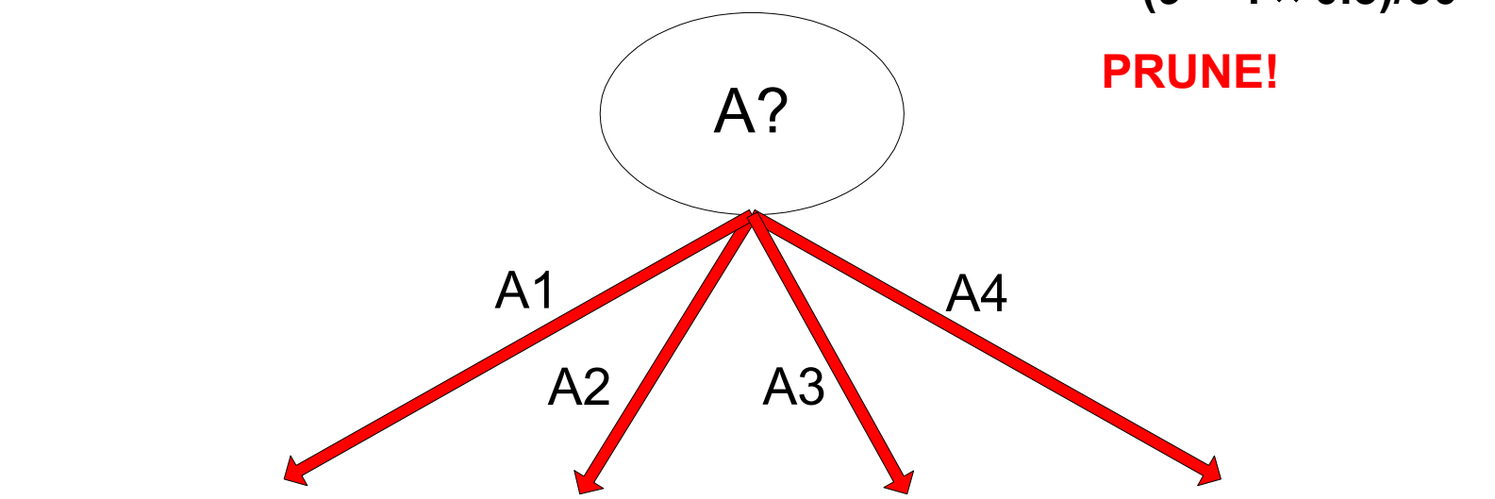
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1