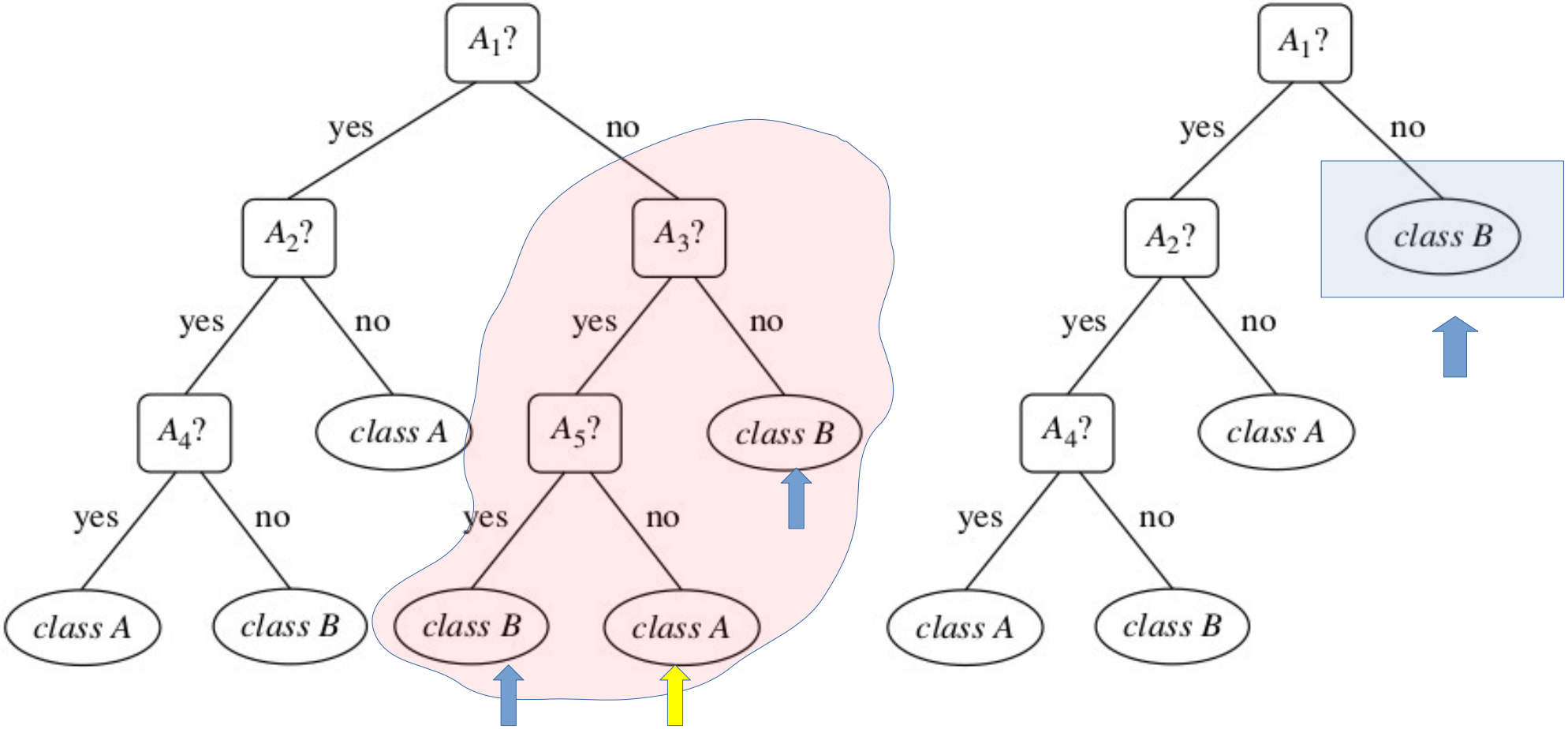
Decision Tree Classification

Overfitting Solution: Postpruning

- Removes subtrees from a “fully grown” tree and replace it with leaf
- The leaf is labeled with the most frequent class among the subtree being replaced



Postpruning: Cost Complexity Pruning

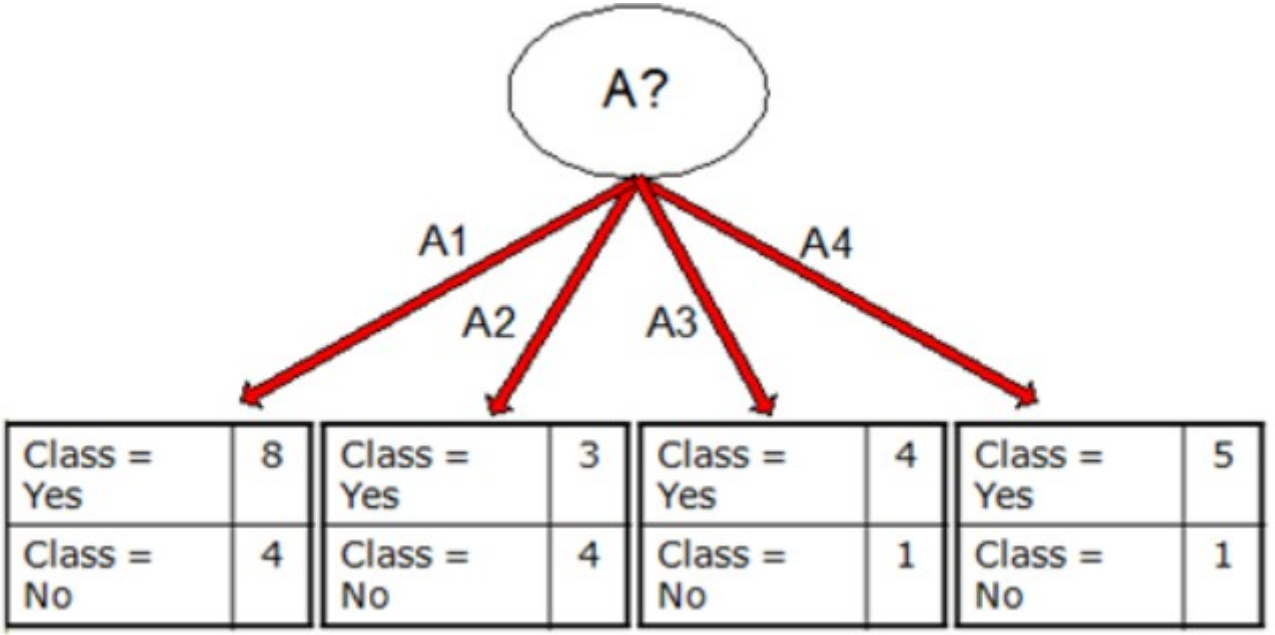
- Used in CART as a postpruning approach.
- It Considers the cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree (where the error rate is the percentage of tuples misclassified by the tree).
- It starts from the bottom of the tree.
- For each internal node, N, it computes:
  - 1 cost complexity of the subtree at N,
  - 2 cost complexity of the subtree at N if it were to be pruned (i.e., replaced by a leaf node).
- The two values are compared. If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept.
- A **pruning set** of class-labeled tuples is used to estimate cost complexity.
- This set is independent of the training set used to build the unpruned tree and of any test set used for accuracy estimation.
- In general, the smallest decision tree that minimizes the cost complexity is preferred.

Postpruning: Pessimistic pruning

- Used in C4.5
- Similar to the cost complexity method in that it also uses error rate estimates to make pruning decision
- It does not require the use of a prune set. Instead, it uses the training set to estimate error rates
- Estimate of accuracy or error based on the training set is overly optimistic and, therefore, strongly biased.
- The pessimistic pruning method therefore adjusts the error rates obtained from the training set by adding a penalty, so as to counter the bias incurred

Pessimistic pruning: Example

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



Training Error:
Without Split: 10/30
With Split: 9/30

- Now to perform pessimistic Pruning we need to add penalty to adjust the error rates obtained from the training set.
- If we add 0.5 penalty per node then:
  - Error(without Split): (10+0.5) /30 = 10.5/30
  - Error(with Split) : (9+ (4*0.5)) /30 = 11/30
- This drop in error suggests that we should prune the node which is split based on attribute A

+ Reasonable training time

+ Fast application

+ Easy to interpret

+ Easy to implement

+ Intuitive

-Not effective for very high dimensional data

Example: words in text classification

–Not robust to dropping of important features