Overfitting Prevention (Pruning) Methods

- Two basic approaches for decision trees
  - Prepruning: Stop growing tree as some point during top-down construction when there is no longer sufficient data to make reliable decisions.
  - Postpruning: Grow the full tree, then remove subtrees that do not have sufficient evidence.
- Label leaf resulting from pruning with the majority class of the remaining data, or a class probability distribution.
- Method for determining which subtrees to prune:
  - Cross-validation: Reserve some training data as a hold-out set (***validation set***, ***tuning set***) to evaluate utility of subtrees.
  - Statistical test: Use a statistical test on the training data to determine if any observed regularity can be dismisses as likely due to random chance.
  - Minimum description length (MDL): Determine if the additional complexity of the hypothesis is less complex than just explicitly remembering any exceptions resulting from pruning.

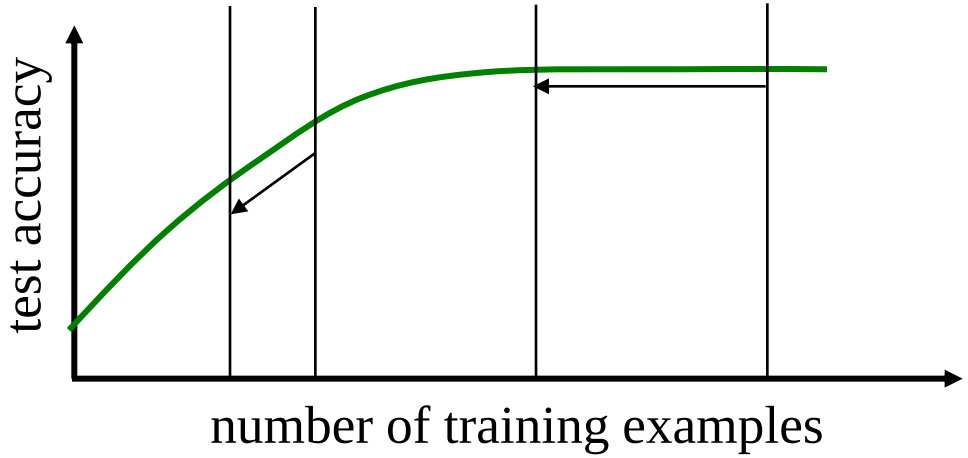
Reduced Error Pruning

- A post-pruning, cross-validation approach.

Partition training data in “grow” and “validation” sets.
Build a complete tree from the “grow” data.
Until accuracy on validation set decreases do:
    For each non-leaf node, n, in the tree do:
        Temporarily prune the subtree below n and replace it with a
            leaf labeled with the current majority class at that node.
        Measure and record the accuracy of the pruned tree on the validation set.
    Permanently prune the node that results in the greatest increase in accuracy on
        the validation set.

Issues with Reduced Error Pruning

- The problem with this approach is that it potentially "wastes" training data on the validation set.
- Severity of this problem depends where we are on the learning curve:



Additional Decision Tree Issues

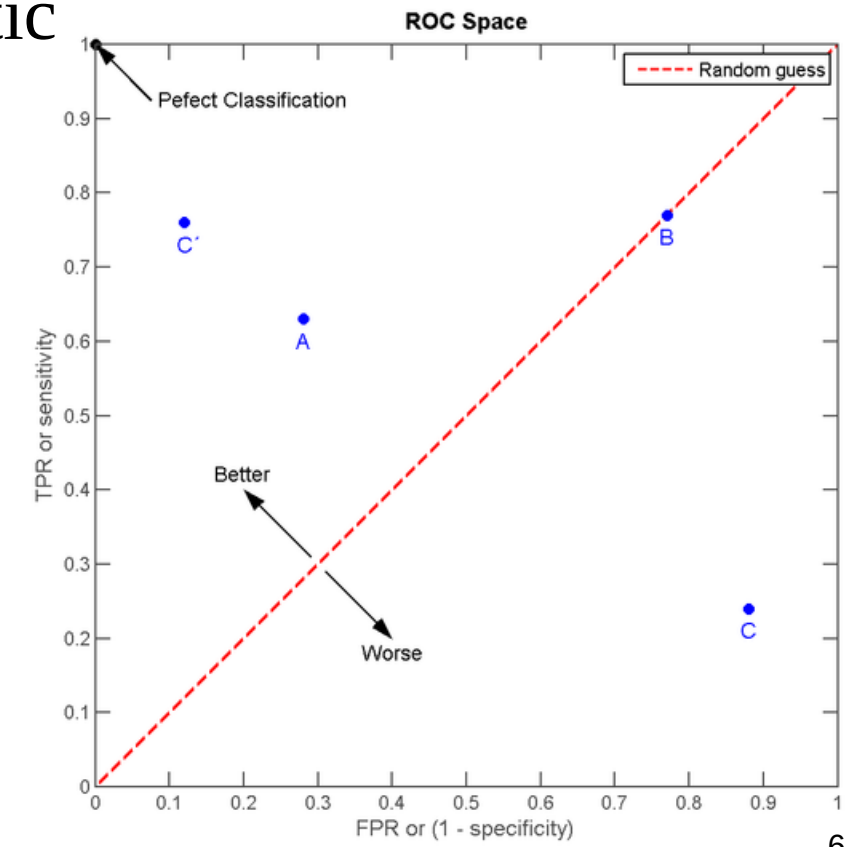
- Better splitting criteria
  - Information gain prefers features with many values.
- Continuous features
- Predicting a real-valued function (regression trees)
- Missing feature values
- Features with costs
- Misclassification costs
- Incremental learning
  - ID4
  - ID5
- Mining large databases that do not fit in main memory

More Terminology

		Test Result	
		Positive	Negative
Ground truth	Positive	True positive	False negative
	Negative	False positive	True negative

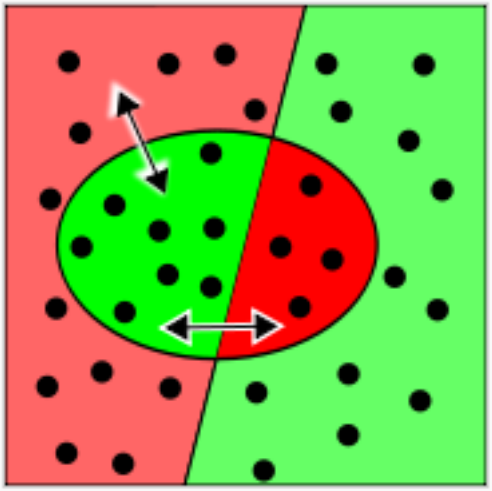
ROC

- ROC: false positives (x-axis) vs. true positives (y-axis)
- receiver operating characteristic (from signal detection theory)



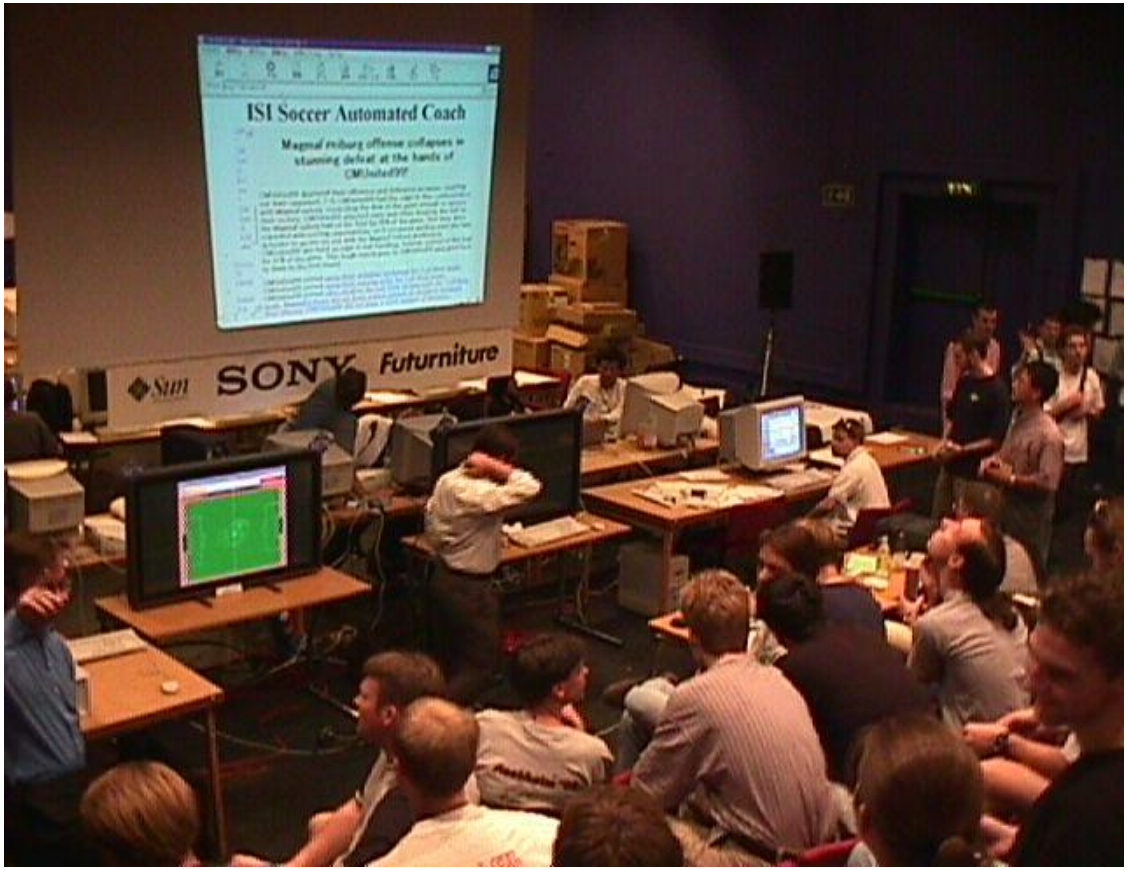
Precision vs. Recall

- Precision = fidelity
  - TP / (TP + FP)
  - 1.0 = Everything retried was relevant
- Recall = exactness
  - TP / (TP + FN)
  - 1.0 = all relevant documents retrieved by search
- F_1 score: harmonic mean
  - 2 * ((P * R) / (P + R))



Recall and precision depend on the outcome (oval) of a query and its relation to all relevant documents (left) and the non-relevant documents (right). The more correct results (green), the better. **Precision**: horizontal arrow. **Recall**: diagonal arrow.

Example Application: ISAAC at RoboCup'99



ISAAC: ISI Soccer Automated Assistant Coach

- ISAAC: analyze synthetic RoboCup teams
  - Analysis of shots on goal, sequences & complete games
- Used extensively before, at (and after) RoboCup'99
  - Past analysis provided surprising results
  - Analyzed behavior of remote teams (over the internet)
  - Game summaries, predictions, analysis at the tournament
- Won RoboCup "Scientific Challenge Award"
- Nair et al, Journal of Agents and Multiagent Systems, 2004

ISAAC's Advice on Goal Kicks

From traces, separate out successful and failed shots on goal

- E.g.,
- Ball velocity = 2.40, shot-aim-point = 3, side = Left, Goal = YES
- Ball velocity = 1.90, shot-aim-point = 10, side = Right, Goal = NO
- Ball velocity = 3.35, shot-aim-point = 5, side = Right, Goal = YES

Use C5.0 to generate rules for offense & defense

Rule: (From Andhill, 2nd place winner in RoboCup '97)

Ball Velocity <= 2.37 m/t
Shot aim point > 6.7 m
-> class No-Goal [0.985] Goals: 0 No Goals: 66

Developer (Andou) "I was surprised … result would help improve Andhill"

LMS: more details

- Chalkboard