Computer Science 477

Overfitting

Lecture 10

Problem Statement

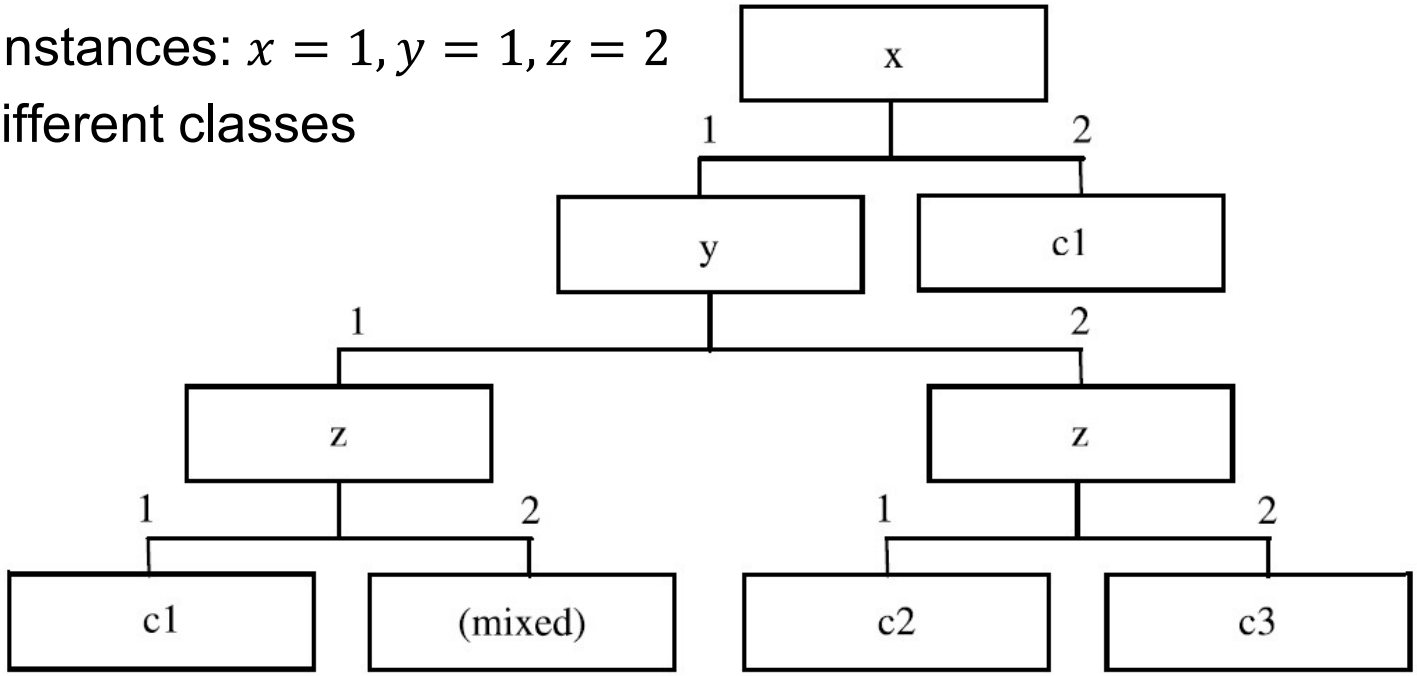
- Classifier suffers from overfitting if it generates a decision tree (or other mechanism) too well adapted to the training set
- Performs well on training set, not well on other data.
- Some overfitting inevitable
- Remedy:
  - Adjust a decision tree while it is being generated
    - Pre-pruning
  - Modify the tree after creation
    - Post-pruning

Digression - Clashes

- Two (or more) instances of a training set have identical attribute values, but different classification.
- Possibilities:
  - Classification incorrectly recorded
  - Recorded attributes insufficient
    - Would need more attributes, normally impossible
- Especially a problem for TDIDT
  - ‘Adequacy condition’
- Must modify basic TDIDT algorithm

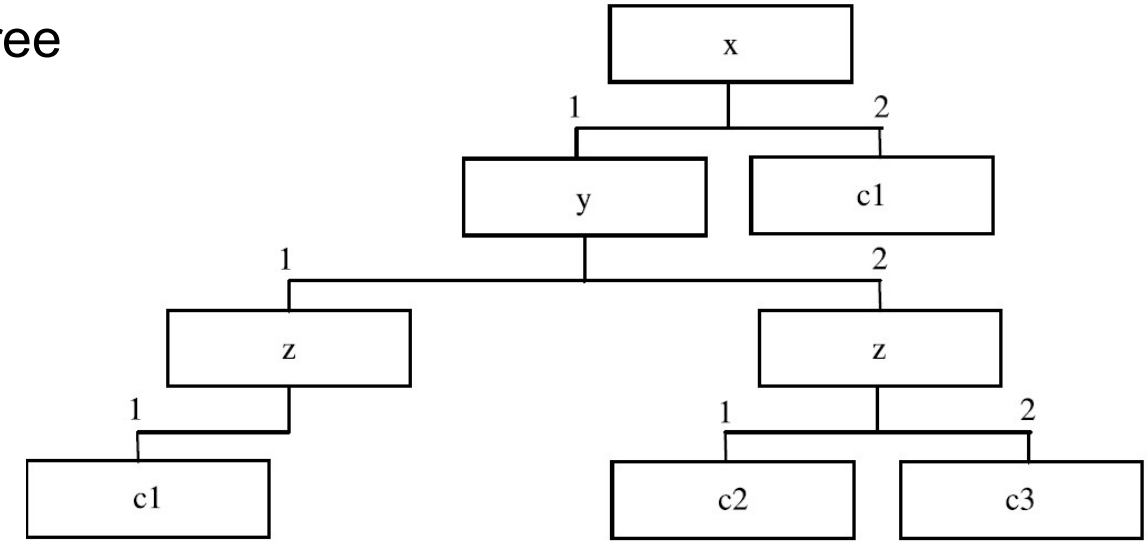
Adapting TDIDT Algorithm

- Clashes in TDIDT algorithm will grow to its greatest length, with instances a lowest nodes having more than one classification.
- Two instances: $x = 1, y = 1, z = 2$
- But different classes



Strategy 1 – Delete Branch

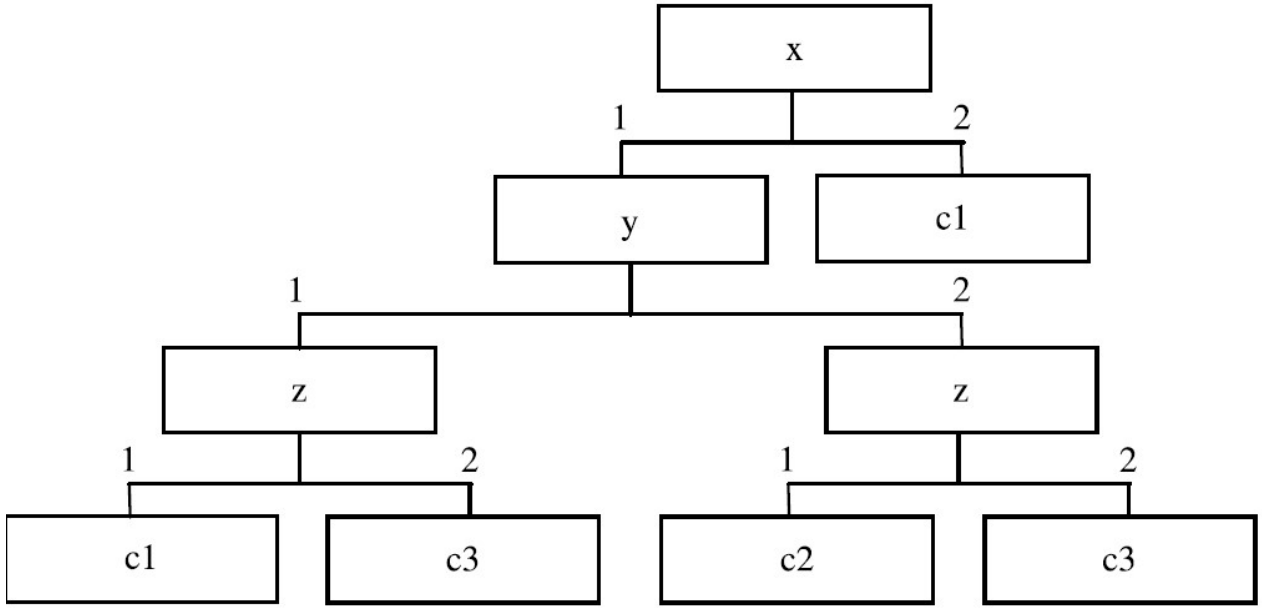
- Discard the branch to the clashing node from the node above
  - Similar to removing to removing clashing instances from the training set.
- Resulting tree



- But cannot classify new instance: $x = 1, y = 1, z = 2$

Strategy 2 : Majority Voting

- Of clashing instances, assume the majority label
  - Similar to changing the label in training data set.
- Assuming c3 is the majority of clashing instances:



Choosing Strategy

- Suppose that 99 instances classified **yes** and one **no**
- Surmise that the one instance was misclassified
  - Change it
- Weather forecasting application:
  - Same attribute values give: 4 rain, 5 snow and 3 fog
- Accept that certain configurations can't be (reliably) classified
- Middle approach: *clash threshold*

Clash Threshold

- Clash threshold – a percentage between 0 and 100.
- Assign all clashing instances the majority class, provided that the instances with the majority class are above the clash threshold.
- Otherwise discard clashing instances.
- Clash threshold = 0 → always assign the majority class
- Clash threshold = 100 → discard class set (delete branch)

Example

- From crx dataset, remove continuous attributes, adequacy condition now violated
- Results with varying thresholds:

Clash threshold	Training set			Test set		
	Correct	Incorr.	Unclas	Correct	Incorr.	Unclas
0% Maj. Voting	651	39	0	184	16	0
60%	638	26	26	182	10	8
70%	613	13	64	177	3	20
80%	607	11	72	176	2	22
90%	552	0	138	162	0	38
100% Del. Branch	552	0	138	162	0	38

- Results for training set in line with results for test set.
- Reducing threshold increases number of correct classifications, but also incorrect classifications
  - Unclassified then tends to 0.

Example continued

- Assign unclassified instances in test set to majority class:

Clash threshold	Training set			Test set	
	Correct	Incorr.	Unclas	Correct	Incorr.
0% maj. voting	651	39	0	184	16
60%	638	26	26	188	12
70%	613	13	64	189	11
80%	607	11	72	189	11
90%	552	0	138	180	20
100% del. branch	552	0	138	180	20

- Highest predictive accuracy on this policy at 70% and 80% threshold

Back to Overfitting

- Typical rule:
  - IF $a = 1$ and $b = yes$ and $z = red$ THEN $class = OK$
- Specializing:
  - IF $a = 1$ and $b = yes$ and $z = red$ and $k = green$ THEN $class = OK$
- Specialized rule covers fewer instances
- Relaxed – fewer constraints on attribute values – increases coverage
- Further down in a decision tree based on fewer and fewer instances
  - Less and less likely to be representative
- Increased specializing (specify) results in a larger rule set
- Standard approach: sacrifice classification accuracy in training set for accuracy in test data, by
  - Pre-pruning (forward pruning)
  - Post-pruning (backward pruning)

Pre-pruning Decision Trees

- If not all instances have the same classification, normally split on the value of a new attribute.
- When pre-pruning, first test whether a termination condition applies.
- If so, current subset is treated as a ‘clash set’
- Resolve by ‘delete branch,’ ‘majority voting,’ etc.
- May reduce accuracy on training set but may be better on test set (and subsequent) data than unpruned classifier.
- Two pruning methods:
  - Size Cuttoff – prune of subset has fewer than X instances
  - Maximum depth: prune if length of branch exceed Y

Examples

- Cutoff at no cutoff, 5 instances, 10 instances

	No cutoff		5 Instances		10 Instances	
	Rules	% Acc.	Rules	% Acc.	Rules	% Acc.
breast-cancer	93.2	89.8	78.7	90.6	63.4	91.6
contact_lenses	16.0	92.5	10.6	92.5	8.0	90.7
diabetes	121.9	70.3	97.3	69.4	75.4	70.3
glass	38.3	69.6	30.7	71.0	23.8	71.0
hypo	14.2	99.5	11.6	99.4	11.5	99.4
monk1	37.8	83.9	26.0	75.8	16.8	72.6
monk3	26.5	86.9	19.5	89.3	16.2	90.1
sick-euthyroid	72.8	96.7	59.8	96.7	48.4	96.8
vote	29.2	91.7	19.4	91.0	14.9	92.3
wake_vortex	298.4	71.8	244.6	73.3	190.2	74.3
wake_vortex2	227.1	71.3	191.2	71.4	155.7	72.2

Examples – Branch Length

- Branch length at 3 and 4

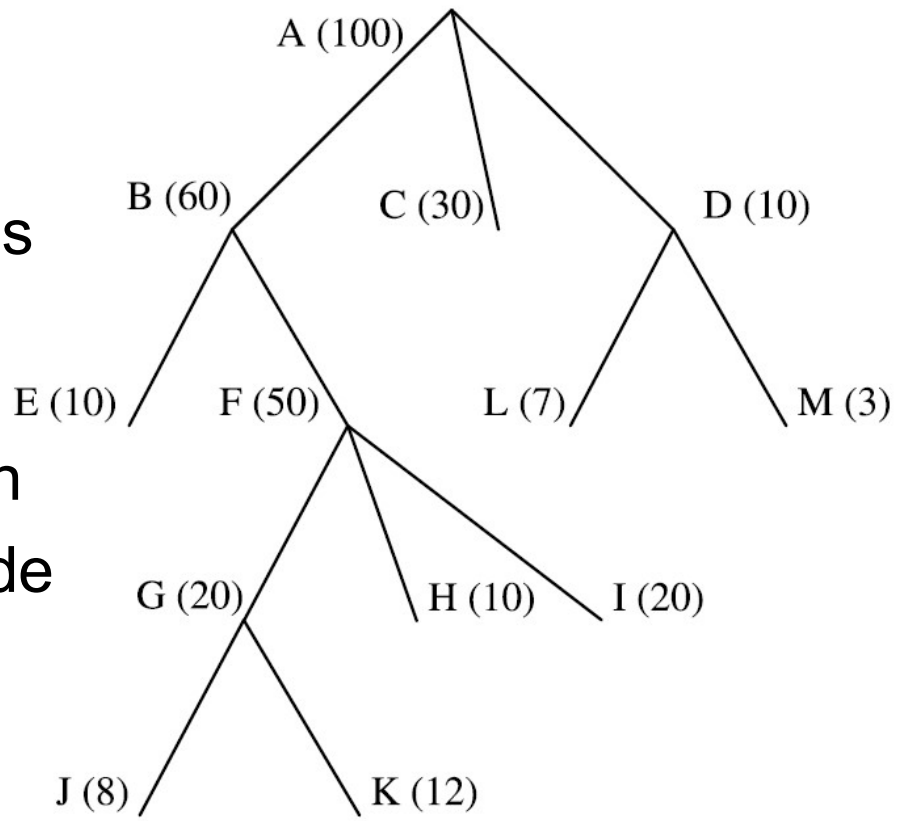
	No cutoff		Length 3		Length 4	
	Rules	% Acc.	Rules	% Acc.	Rules	% Acc.
breast-cancer	93.2	89.8	92.6	89.7	93.2	89.8
contact_lenses	16.0	92.5	8.1	90.7	12.7	94.4
diabetes	121.9	70.3	12.2	74.6	30.3	74.3
glass	38.3	69.6	8.8	66.8	17.7	68.7
hypo	14.2	99.5	6.7	99.2	9.3	99.2
monk1	37.8	83.9	22.1	77.4	31.0	82.2
monk3	26.5	86.9	19.1	87.7	25.6	86.9
sick-euthyroid	72.8	96.7	8.3	97.8	21.7	97.7
vote	29.2	91.7	15.0	91.0	19.1	90.3
wake_vortex	298.4	71.8	74.8	76.8	206.1	74.5
wake_vortex2	227.1	71.3	37.6	76.3	76.2	73.8

Post-Pruning - Definition

- Begin by generating complete tree subsequently adjusting to improve accuracy
- Two Methods:
  - Begin by converting tree to equivalent set of rules
    - Addressed later
  - Replace some subtrees by leaf nodes, according to calculated estimates of error rates
  - Four of many variants:
    - Reduced Error Pruning
    - Pessimistic Error Pruning
    - Minimum Error Pruning
    - Error Based Pruning

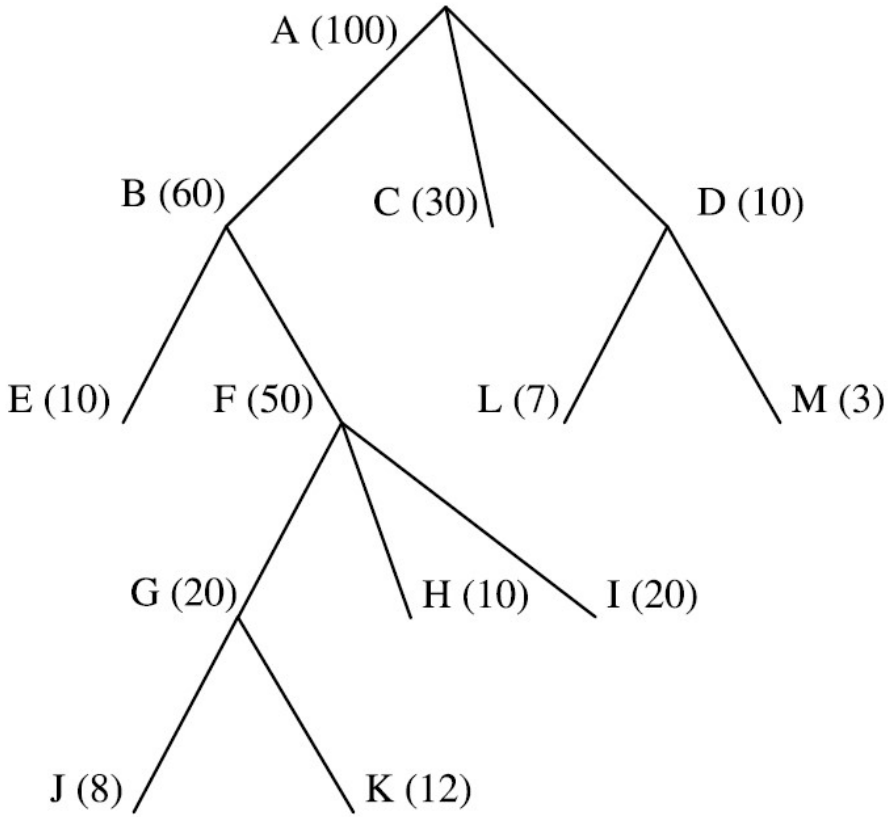
Example

- A complete tree
- Number at each node denotes number of training instances covered by it
- Leaves have the classification
- The branch from A to leaf node J corresponds to a decision rule
- A decision path from A to G (rather than A to J and A to K) forms an incomplete rule.



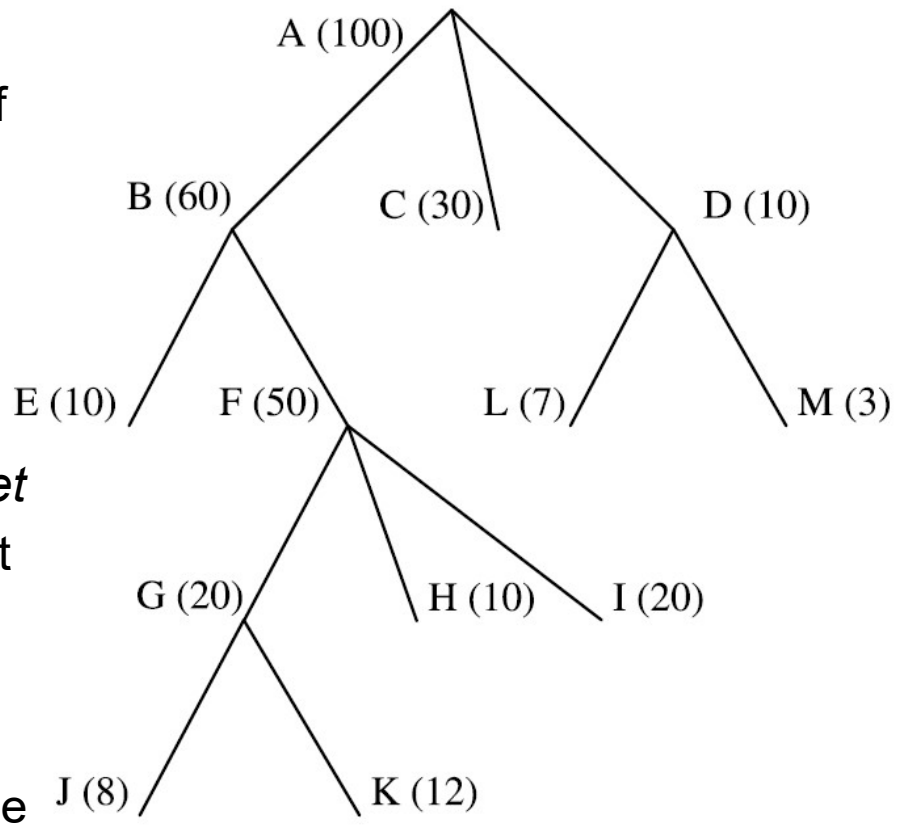
Example

- When post-pruning, look for a non-leaf nodes that have descendants of length 1.
- In this tree, only node G and D are candidates for pruning (consolidation).
- If the pruning condition is met, subtree hanging from a node is replaced by the node itself
- Working from the bottom ft the tree upwards
  - Prune one subtree at a time.
  - Continues until no further pruning is justified



Example

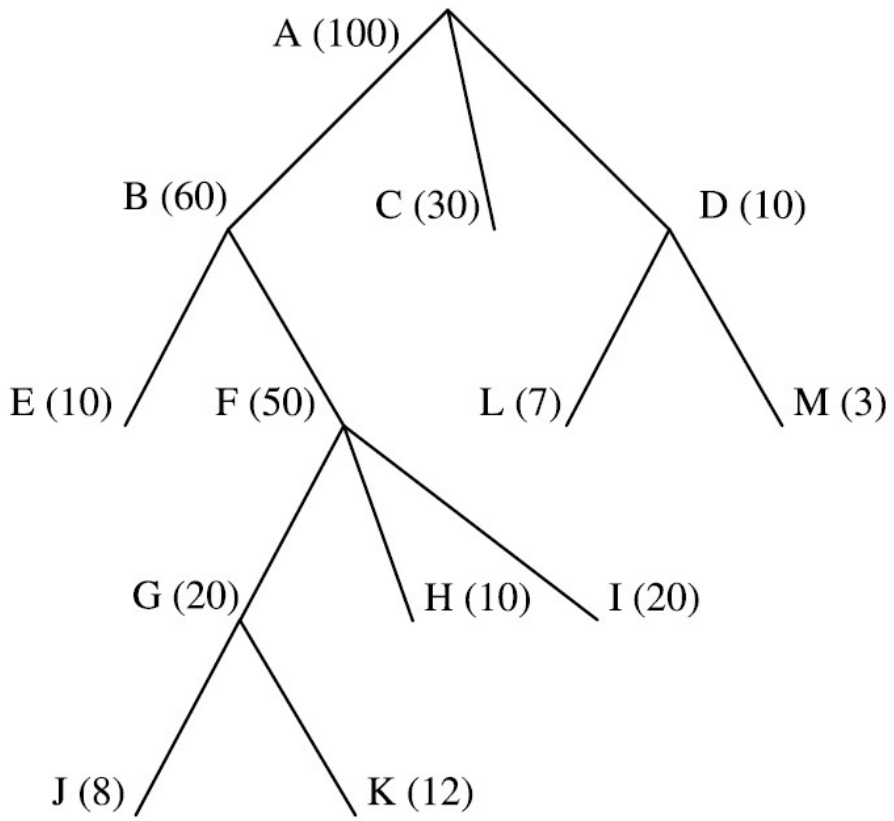
- Start by considering the replacement of the subtree G, J, K with G itself
- Question: how does the error rate at G compare with the error rate of the complete rules, ending at J and K?
- Need a way to estimate error rate.
- Count misclassifications in a *pruning set*
  - Distinct from *training set* and distinct from *test set*.
- Alternative: use a formula to estimate errors
  - Might use number of instances at the node belonging to each class
  - Prior probability of each class
  - Reduced Error Pruning, Pessimistic Error Pruning, Minimum Error Pruning, Error Based Pruning



Example

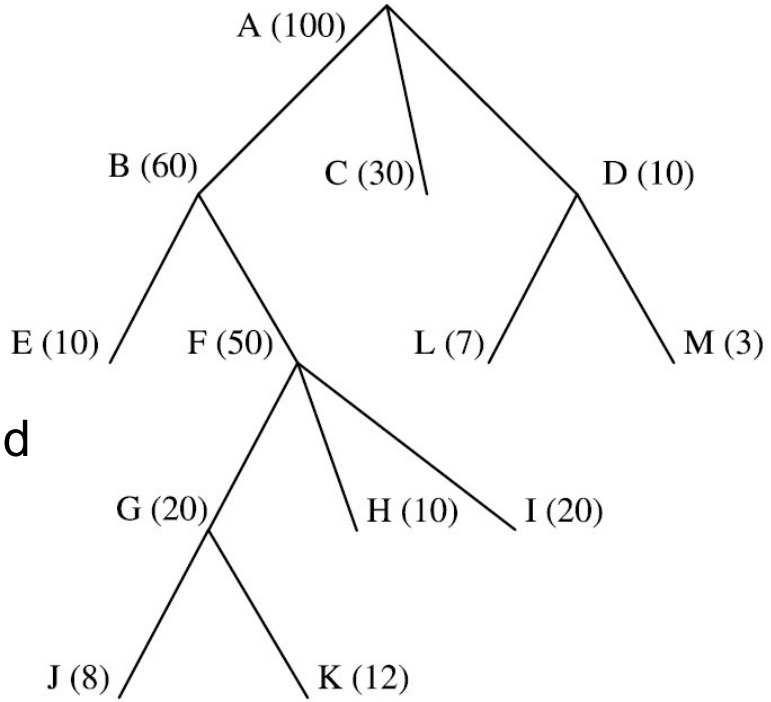
- Estimated error rates for full tree:

Node	Estimated error rate
A	0.3
B	0.15
C	0.25
D	0.19
E	0.1
F	0.129
G	0.12
H	0.05
I	0.2
J	0.2
K	0.1
L	0.2
M	0.1



Example

J	0.2
K	0.1



- Estimated error rates at J and K are 0.2 and 0.1
  - Some measure or other
- Correspond to 8 and 12 instances
- To combine their error rate, computer their weighted average:

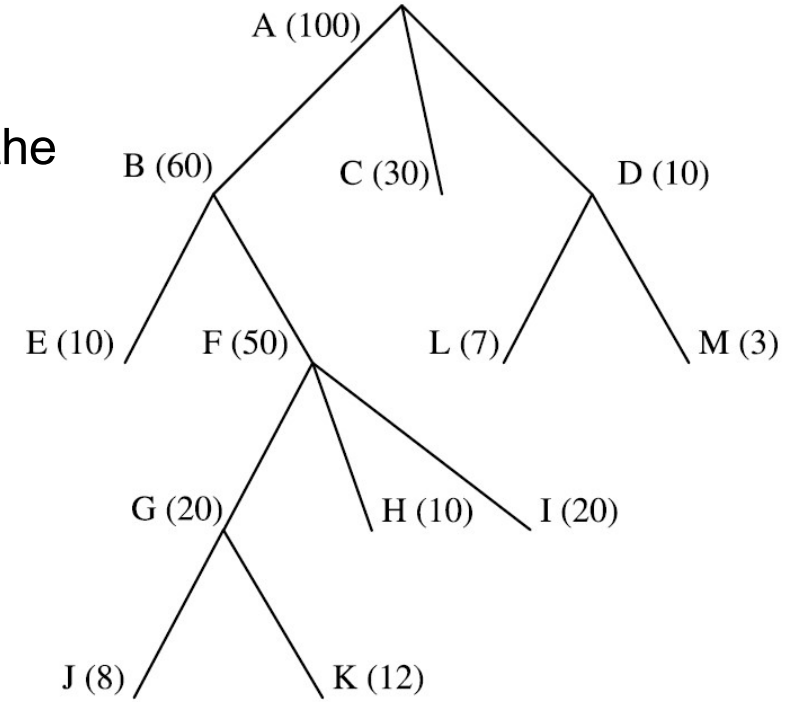
$$\frac{8}{20} \times 0.2 + \frac{12}{20} \times 0.1 = 0.14$$

- Called *backed-up estimate*
- Compare this value for the error rate estimated for G, 0.12.
  - Called the *static error rate*

G	0.12

Example

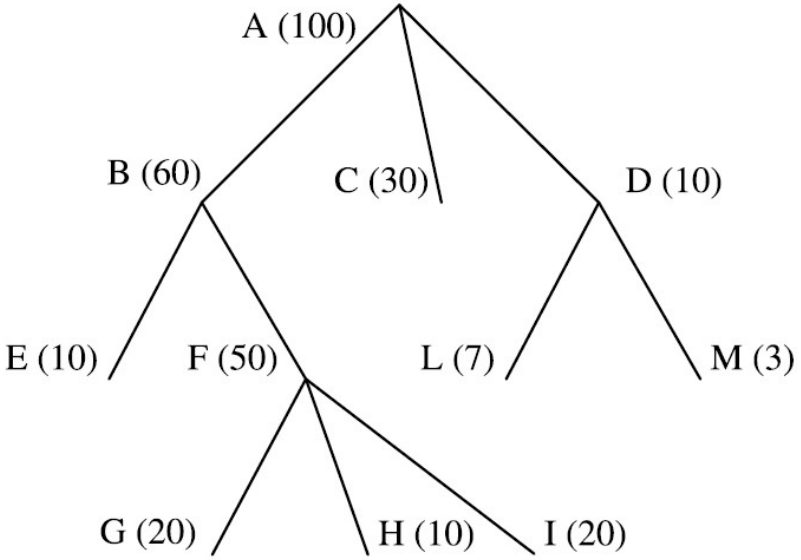
- The static rate of error (0.12) is less than the backed up rate (0.14)
- So the J-k subtree is pruned



F	0.129
G	0.12
H	0.05

Example

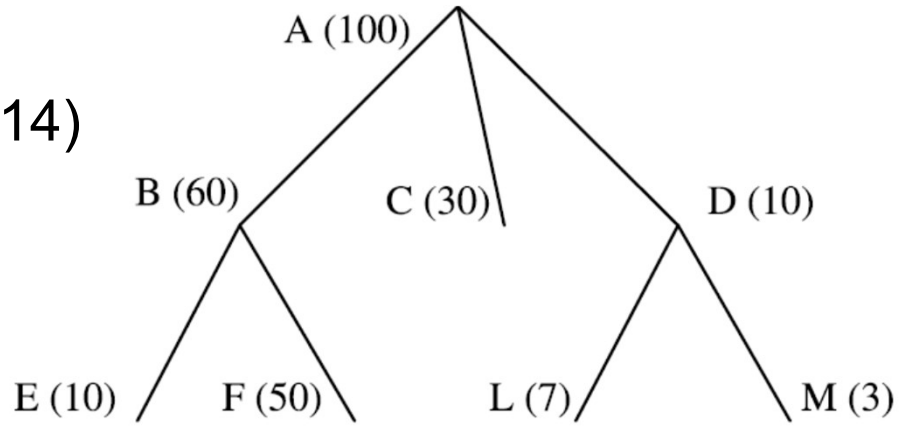
- The static rate of error (0.12) is less than the backed up rate (0.14)
- So the J-k subtree is pruned
- Next consider whether it is beneficial to prune the subtree below F.
- The static error rates at nodes G, H and I are 0.12, 0.05 and 0.2.
- Backed-up error rate at node F is $(20/50) \times 0.12 + (10/50) \times 0.05 + (20/50) \times 0.2 = 0.138$
- Static error rate at node F is 0.129, smaller than 0.138
- So we prune it.



E	0.1
F	0.129
G	0.12
H	0.05
I	0.2

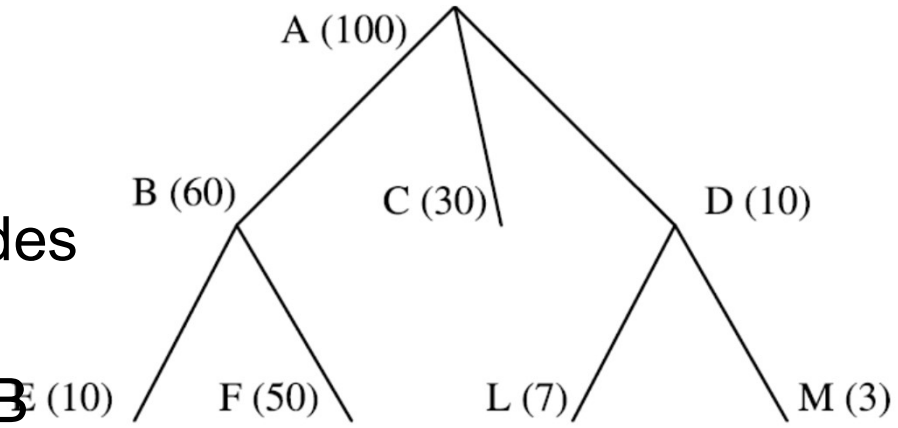
Example

- The static rate of error (0.12) is less than the backed up rate (0.14)
- So the J-k subtree is pruned
- Next consider whether it is beneficial to prune the subtree below F.
- The static error rates at nodes G, H and I are 0.12, 0.05 and 0.2.
- Backed-up error rate at node F is $(20/50) \times 0.12 + (10/50) \times 0.05 + (20/50) \times 0.2 = 0.138$
- Static error rate at node F is 0.129, smaller than 0.138
- So we prune it.



Example

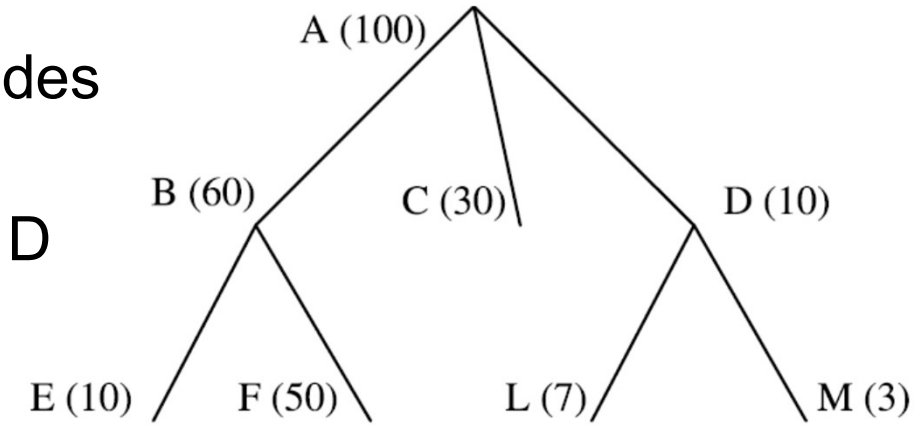
- Candidates for pruning are subtrees from nodes B and D
- For B; Static error rates at nodes E and F are 0.1 and 0.129
- Backed-up error rate at node B is $(\frac{10}{60}) \times 0.1 + (\frac{50}{60}) \times 0.129 = 0.124$
- Static error rate at B is which is 0.15.
- Splitting at B reduces error rate, so we do not prune the subtree
  - $0.15 > 0.12$



A	0.3
B	0.15
C	0.25
D	0.19
E	0.1
F	0.129
C	0.12

Example

- For D; Static error rates at nodes L and M are 0.2 and 0.1
- Backed-up error rate at node D is $(\frac{7}{10}) \times 0.2 + (\frac{3}{10}) \times 0.1 = 0.17$
- Static error rate at D is 0.19.
- Splitting at D reduces error rate from 0.19 to 0.17 so we do not prune the subtree



D	0.19

L	0.2
M	0.1