CIS501 – Lecture 10

Woon Wei Lee
Fall 2013, 10:00-11:15am,
Sundays and Wednesdays

Masdar INSTITUTE

For today:

- Administrative stuff
  - Project
- Information Gain revisited
  - Tree induction using IG
  - Alternative Scoring Schemes
- Tree Pruning
- Presentations

Information Gain (Cont'd)

- The story so far..
  - Basic decision tree framework → branch on attributes sequentially, i.e. "divide and conquer"
  - Question: which attribute to use first? Different ordering can result in wildly different trees
  - Information Theoretic solution in the form of Information Gain (or IG), defined as:

$$IG(a) = H(X) - H(X|a)$$

- Defines the degree to which a feature is "informative" about the class label
  - Characterized by the reduction in uncertainty or Entropy in the resulting instance subsets
  - Aside: also, useful for feature selection in general classification tasks..

Tree induction using IG

<u>Revised "TDIDT" algorithm (Known as the ID3 algorithm):</u>

- Choose the root node to be the attribute a with the highest information gain relative to X.
- For each value *v* that *a* can possibly take, branch the node.
- For each branch from A corresponding to value *v*, calculate X_v.
  - If X_v is empty, choose the category $c_{default}$ which contains the most examples from X, and put this as the leaf node category which ends that branch.
  - If X_v contains only examples from a category *c*, then put *c* as the leaf node category which ends that branch.
- Otherwise, remove *a* from the set of attributes which can be put into nodes.
- Insert new node in the decision tree
  - Use attribute with the highest information gain *relative to* X_v (important: *not* X).
- Recurse...

Alternative Scoring Schemes

- Information Gain has already been used very fruitfully in the ID3 algorithm.
  - However, it is not the only, or necessarily the best means of feature ranking...
- For e.g., it is *biased* in favour of multi-valued features.
- The larger the number of possible values *v* that a feature can take, the higher the IG is likely to be.
- For our friend the golf dataset:
  - Imagine that we had an additional feature, “ID”
  - Obviously no causal relationship whatsoever with play/noplay
  - Yet, it would score extremely high (or, perfect score, even) for IG
  - Could occur commonly in practice, for e.g. customer's CC details..

ID	Outlook	Windy	Class
a	Sunny	TRUE	don’t play
b	Sunny	TRUE	don’t play
c	Rain	TRUE	don’t play
d	Rain	TRUE	don’t play
e	Sunny	FALSE	play
f	Sunny	FALSE	play
g	Overcast	TRUE	play
h	Overcast	FALSE	play
i	Overcast	TRUE	play
j	Overcast	FALSE	play
k	Rain	FALSE	play
l	Rain	FALSE	play
m	Rain	FALSE	play

Gain Ratio

- Need to “normalize” IG w.r.t. to the number of values that a feature can take.
- Quinlan (inventor of ID3), suggests the use of the *SplitInfo statistic,* defined as:

$$P(X,a)=\sum_{v}\frac{|X_v|}{|X|}\log_2\frac{|X_v|}{|X|}$$

- Helps to account for the increase or gain in entropy resulting from the partitioning itself.
- This is combined with Information Gain, to form the “Gain Ratio”:

$$G(X,a)=\frac{IG(X,a)}{P(X,a)}$$

- Used with the famous C4.5 Decision Tree induction algorithm (discussed later)

Gini Impurity Index

- This is another scoring function which can be used to rank features.
- Is determined for the instances in a given set or subset
- Defined as the probability that a randomly selected instance is wrongly classified based on a label randomly sampled from that subset

$$I_G(f) = \sum_{1}^{m} f_i(1 - f_i)$$
$$= \sum_{1}^{m} f_i - \sum_{1}^{m} f_i^2 \quad = 1 - \sum_{1}^{m} f_i^2$$

- f_i is the proportion of instances in the set which are from class *i*.
- A measure of *impurity* (i.e. undesirable!)
  - Hence, an attribute which splits the instances into more homogeneous subsets will have a lower Gini index.
  - **Question: If only one class present, what is the I_G?**
- This is for a single subset.. to score an attribute, evaluate I_G for each of the branches, and calculate weighted average as with IG
- Used with the CART algorithm.

Tree Pruning

- A further issue which confounds decision tree design is the need to avoid *overfitting*.
- For e.g., ID3 will result in continuous repetitions of the induction algorithm until uniform leafs are reached
    - In general, constraints encourage parsimony (Remember Occam's Razor)
    - In practice this often results in overly-complex decision trees..
    - Each branch results in additional and over-specialization
    - If nothing else, this also reduces the “elegance” or comprehensibility of a tree
- One solution is to reduce the size of a tree so that overfitting is avoided.
- We will consider two approaches to achieving this:
    - Pre-pruning
    - Post-pruning

Pre-Pruning

- Also known as “forward pruning” - as might be anticipated, involves the prevention of excessive branching rather than explicit removal.
- General approach – incorporate some *termination criterion* (other than uniform class labels)
- Examples:
  - Minimum Size – Terminate recursion if size of instance subset falls below pre-defined threshold
  - Maximum Depth – Limit the depth of the tree/number of branches allowed
- Terminated branches will be replaced with a leaf labelled by *majority vote*.
- Pruned tree will always have a higher training error, but will often give better performance when applied to the test set.

(Figures on following pages help illustrate...)

	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

Pruning with subset size termination criterion
(Source: *Principles of Data Mining, Max Bramer*)

Masdar INSTITUTE

	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

Pruning with tree depth limit termination criterion
(Source: *Principles of Data Mining, Max Bramer*)

	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

Pruning with tree depth limit termination criterion
(Source: *Principles of Data Mining, Max Bramer*)

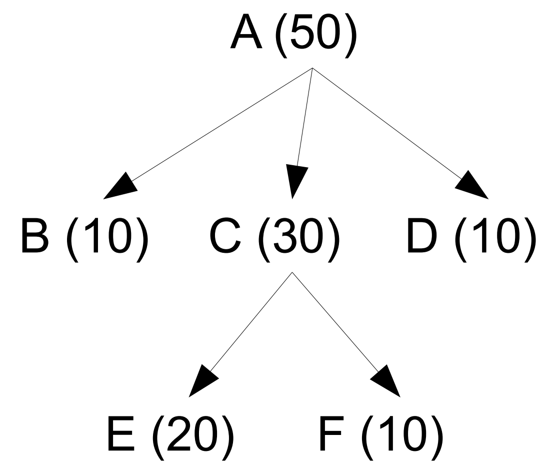
Masdar INSTITUTE

(Cont'd)

- Notes: results in tables obtained using:
  - 10-fold cross validation
  - Majority voting on pruned branches
- Observations:
  - In most cases, pruning resulted in at least some improvement in accuracy
  - In all cases number of decision rules required was reduced, in some cases dramatically
  - Pruning on subset size gave modest improvements in accuracy but accompanied by reductions in tree size
  - Pruning on tree depth seemed to give results that not as accurate, but with bigger reductions on tree sizes
- However, these are sweeping statements – can be quite difficult to generalize
- In practice, selection of pruning technique, as well as appropriate thresholds would be optimized w.r.t. data sets with cross-validation.

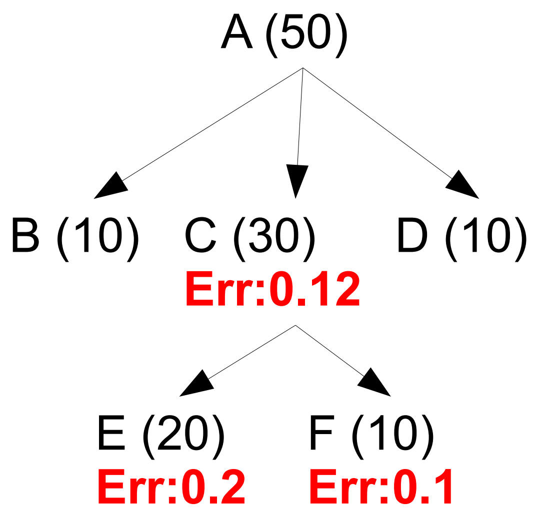
Post-pruning

- Also known as “backward pruning” - i.e. nodes and subtrees are removed after full tree has been constructed
- Computationally more intensive than pre-pruning, but often produces better results.
- Numerous variants but generally based on reducing the final *error* of the decision tree
- Basic idea - for tree on the right:
    - Numbers in brackets are number of instances in that node..
    - Each node to which **only leaves are attached** is considered for pruning (in this case only node *C*)
    - Idea is to evaluate combined error rate of nodes *E* and *F*
    - This is compared with error rate of node *C*, and if greater, replace node C with a leaf.



Post-pruning (Cont'd)

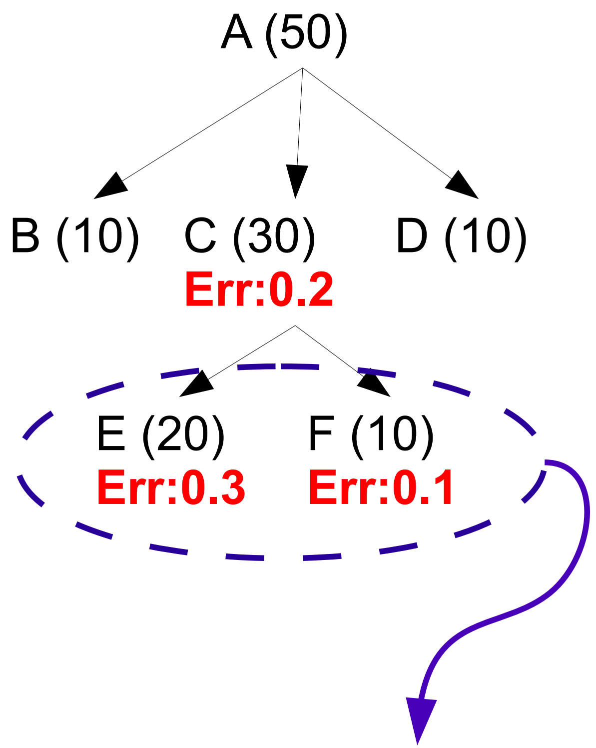
- Numerous variants but generally based on reducing the final *error* of the decision tree
- Basic idea - for tree on the right:
  - Numbers in brackets are number of instances in that node..
  - Each node to which **only leaves are attached** is considered for pruning (in this case only node *C*)
  - Idea is to evaluate combined error rate of nodes *E* and *F*
  - This is compared with error rate of node *C*, and if greater, replace node C with a leaf.
  - “True” error rate can be estimated, for e.g. by using a “pruning set” (separate from the training and test sets)
  - Assume that error rates are as shown on right..



Masdar INSTITUTE

Post-pruning (Cont'd)

- Basic idea - for tree on the right:
    - Numbers in brackets are number of instances in that node..
    - Each node to which **only leaves are attached** is considered for pruning (in this case only node *C*)
    - Idea is to evaluate combined error rate of nodes *E* and *F*
    - This is compared with error rate of node *C*, and if greater, replace node *C* with a leaf.
    - “True” error rate can be estimated, for e.g. by using a “pruning set” (separate from the training and test sets)
    - Assume that error rates are as shown on right..
    - “Backed-up” error exceeds the “static” error for node *C*, hence nodes *E* and *F* are duly pruned..
    - Node *C* is replaced with a leaf



$$Err(E,F) = \frac{2}{3} \times 0.3 + \frac{1}{3} \times 0.1$$

$$= 0.23 > Err(C)(0.2)$$

Error estimation

- The whole procedure is repeated until no further instances of "prunable" leaves are found
- In general, this procedure works well to reduce error in final test, however:
  - Holding out a separate pruning set is wasteful
  - An alternative approach is to try to estimate this true error rate from the training error at each of this nodes
  - Confidence bounds for the training error is found using Error Function
  - Upper limits of this error are used
- Above procedure is known as "pessimistic" pruning, and is used in the C4.5 Decision Tree induction algorithm.

Masdar INSTITUTE

Cost-Complexity pruning

- A further enhancement to previous procedure is employed by the ***CART*** algorithm
- The algorithm includes an additional cost term which explicitly seeks to reduce the size of the tree
- The idea is to find a series of trees that minimize the following cost function:

$$CC(T)=Err(T)+\alpha|T|$$

- Where
  - *CC(T)* is the *Cost-Complexity* of tree *T*
  - *Err(T)* is the total error rate of the tree T, and is calculated using the training data
  - $|T|$ is the number of nodes in the tree
  - α is a weighting factor which determines the emphasis on $|T|$

Masdar INSTITUTE

Cost-Complexity pruning

- Broadly speaking, the algorithm proceeds as follows:

1. Start with *α=0 (*no nodes are pruned)
2. *α* is increased gradually, until *CC(T)* exceeds that for a subtree, which is retained.
3. Procedure is repeated until only a single node is left, leaving a sequence of (sub)trees

- Once complete, the error rate for each of the trees in the sequence are evaluated using a separate validation set.
- Tree with smallest validation error is retained, known as the “minimum error tree”.