Weekly report of lessons

Name: Mayank Kumar
Roll No: 19CS30029
The week: 30-08-2021 to 05-09-2021

The topics covered:

Information Gain: Gain Function; Hypothesis Space Search: ID3 Algorithm, Hypothesis Space Search in Decision Trees, Hypothesis Space Search by ID3, Restriction bias vs Preference bias, Inductive bias in ID3; Overfitting: Avoiding; Evaluating subtrees to prune: Reduced-Error Pruning, Rule Post-pruning; Extension of basic algorithm: Continuous valued attributes, Candidate thresholds, Attributes with many values, Unknown Attribute Values, Attributes with costs, Gini Index; Regression Tree: Rectilinear Division, Growing Regression Trees, Regression Tree Pruning; Usefulness of Decision Trees; Learning rules directly: Sequential covering algorithms, General to specific beam search; Oblique Decision Tree; Random Decision Forest

Summary topic wise:

Information Gain: The decrease in entropy when a dataset is split on an attribute.
For constructing a decision tree, we have to find attribute that returns highest information gain.
Gain Function: The measure of the reduction of uncertainty; value lies between 0 and 1

Hypothesis Space Search: Search complete hypothesis space; Target function is included there
ID3 Algorithm: For the input dataset S, split S into subsets and make a decision tree using best attribute A. If S is empty, return by adding a leaf node below with most frequent label. Recursively call ID3 with S and set of remaining attributes.

Hypothesis Space Search by ID3: Search space of possible decision trees by hill-climbing approach on information gain. It uses all training examples at each step.
Restriction bias vs Preference bias: Restriction bias has incomplete hypothesis space, while Preference bias has incomplete search strategy.
Inductive Bias in ID3: Preference for smaller trees

Overfitting: A hypothesis h is overfitting if there exists an hypothesis h' such that h has smaller training error but greater test error than h'.
Avoiding Overfitting: Pre-pruning - stop growing tree when data isn't enough to make reliable choice; Post-pruning - grow full tree and remove nodes having insufficient evidence

Evaluating subtrees to prune: Methods: Cross-validation, Statistical Testing, Minimum description length
Reduced-Error Pruning: Cross-validation approach prunes tree to increase validation set accuracy the most.
Rule Post-pruning: Tree is grown until best fit and then converted to equivalent set of rules. Each rule is independently pruned and final rules are sorted into desired sequence.

Extension of basic algorithm:
Continuous valued attributes: Create discrete intervals and consider them as discrete values
Candidate thresholds: Generated at places where target value changes quickly and the attribute associated with maximum information gain is chosen.
Attributes with many values: Instead of Gain, use GainRatio.
For dataset S and any attribute A, $\boldsymbol{GainRatio(S,A) = Gain(S,A) / SplitInformation(S,A)}$
$\boldsymbol{SplitInformation(S,A) = -\sum_{i=1}^{c} |S_i/S| \log_2 |S_i/S|}$
S_i -> subset of S for which A has value v_i

Unknown Attribute Values: For missing values, either assign most common value of attribute A among examples sorted to node n, or assign most common value of A among examples with same target value, or assign by using probability p_i of each possible v_i of A.
Attributes with costs: Replace gain by $\boldsymbol{(Gain^2(S,A) / Cost(A))}$ or $\boldsymbol{((2^{Gain(S,A)} - 1) / (Cost(A) + 1)^{\omega})}$ where $\omega \in [0, 1]$ determines importance of cost

Gini Index: For class i with probability p(i), $\boldsymbol{Gini = \sum_i p(i)(1 - p(i)) \Rightarrow Gini = 1 - \sum_i p(i)^2}$
After applying attribute A, the resulting Gini index is
$\boldsymbol{Gini(A) = \sum_{v \in A} p(v) \sum_i p(i|v) (1 - p(i|v))}$

Regression Tree: Attribute space is partitioned into set of subspaces to predict output. Splits are chosen to minimize sum of squared errors.
Rectilinear Division: Regression tree is a piecewise constant function of the input attributes

Growing Regression Trees: Average output of the learning cases reaching a leaf is prediction at that leaf. Impurity $\boldsymbol{I(LS) = var_{y|LS}\{y\} = E_{y|LS}\{(y - E_{y|LS}\{y\})^2\}}$, where *var* is variance, *E* is expectation and *LS* is learning space. The best split reduces variance the most.
Regression Tree Pruning: Pre-pruning and Post-pruning can be applied here too. In post-pruning, tree minimizing squared error on VS is selected.

Usefulness of Decision Trees: Handles huge datasets very fast, Flexible with different problems, Interpretability

Learning rules directly: A set of rules can be derived. If <antecedent> then <consequent>.

Sequential covering algorithms: Learn rules one by one involves learning rule and removing the examples covered and iterating the process to cover all the examples.
General to specific beam search: Only k top-performing nodes are chosen which are more probable to form a rule; used by learn-one-rule to find the one with max accuracy.

Oblique Decision Tree: It checks the combination of attributes at a node and the hyperplane divides two halves of space.
Random Decision Forest: Forest formed of trees from randomly constructed subspaces.

Concepts challenging to comprehend:
It was difficult to understand the algorithm behind General to specific beam search at first sight.

Interesting and exciting concepts:
The idea behind decision trees continue to impress as this time, we learned to use the same concept to predict the values close to functional value just by changing the way we split the tree from Information Gain to Sum of Squared Errors

Concepts not understood:
None.

Any novel idea of yours out of the lessons:
The way in which Sequential covering algorithms works is it provides an efficient, greedy algorithm from learning but if there are multiple target classes, it would be better to start with the least frequent class, learn a rule for it, remove all covered instances and move on to next least frequent one. This will increase the coverage of this algorithm as now, it takes proper care of less common attributes, and would also make it more robust against monotonic transformations of the input features.
