Supervised Learning

Decision Trees and Forest

Dr. Neeta Nain

Department of Computer Science and Engineering
Malviya National Institute of Technology, Jaipur

2020

Outline

Parametric vs Non Parametric

- Parametric reduces the problem of estimating a pdf, discriminant, or regression function to estimating the values of a small no of parameters. This assumption does not always hold and we may incur a large error if it does not
- In Non parametric we assume that similar inputs have similar outputs, the world is smooth and functions - densities, discriminants, or regression change slowly. Alg finds the similar past instance from training set using a suitable distance measure and interpolate them for the right answer
- Parametric - all of the training instances affect the final global estimate, in Non parametric - there is no single global model; local models are estimated as they are needed, affected only by the nearby training instances

Parametric vs Non Parametric

- Non parametric - do not assume any a priori parametric form for the underlying densities, it is not fixed but its complexity depends on the size of the training set, or the complexity of the problem inherent in the data
- Non parametric are also called instance based or memory based learning algs, the training instances should be stored and requires memory $O(N)$, further given an input similar ones should be found - requires computation time of $O(N)$. Lazy learning algs, as they do not compute a model with the training set but postpone the computations until a test instance is given
- Parametric - are eager learners, model is quite simple and has a small no of parameters, of $O(d)$, or $O(d^2)$, once these parameters are computed from the training set, keep the model only. $N >> d$, (d^2) the increased need for memory and computation is the disadvantage of the non parametric methods

Decision Tree Classifier

- DT were made poular in statistics by Breiman et al. 1984 and in machine learning by Quinlan 1986 and Quinlan 1993
- A predictive model which maps observations about an item to conclusions about the item's target value
- Simple, hierarchical tree structure, which performs supervised classification using divide-and-conquer strategy
- Comprises a directed branching structure with a series of question. Questions are placed at the decision nodes; each tests the value of a particular attribute (feature) of the pattern (object) and provides a binary or a multi-way split.
- Starting node is known as the root node, branches corresponds to the possible answers (attribute values), successive decision nodes are visited until a terminal or leaf node is reached or decision is made (class is assigned)
- Classification is performed by routing from the root node until arriving at the leaf node

A Three-Level Decision Tree

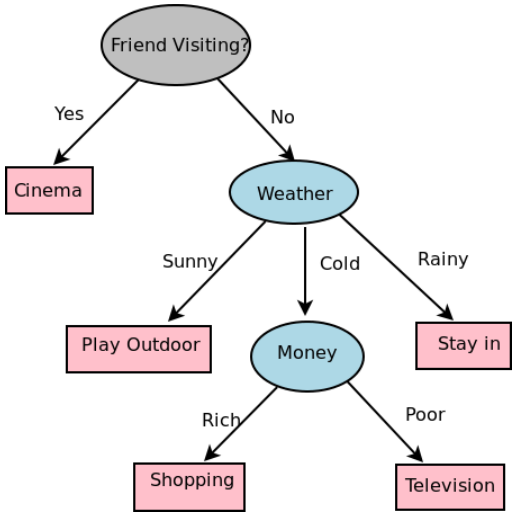


Figure: A three-level decision tree for determining what to do on a Sunday morning.

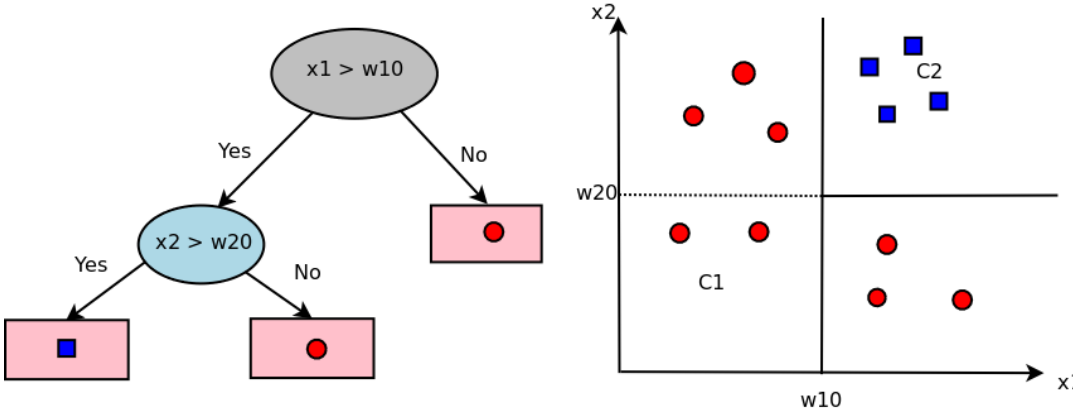
- When target variable take a finite set of values - classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels
- Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees

Decision Tree Properties

- More general than other representations, easy to use
- Can be used with non metric/categorical data, including nominal data with no natural ordering
- Clear interpretability, providing a natural way to incorporate prior knowledge. Straightforward to convert the tests into logical expressions
- Once contructed, very fast since they require very little computations
- A DT is a heirarchical model for supervised learning where the local region is identified in a sequesnce of recursive splits in a no of smaller steps
- It is a nonparametric model, we do not assume any parameter form for the class densities and the tree structure is not fixed apriori but grows, branches and leaves are added, during learning
- Trees are frequently used in computer science to decrease complexity from linear to log time

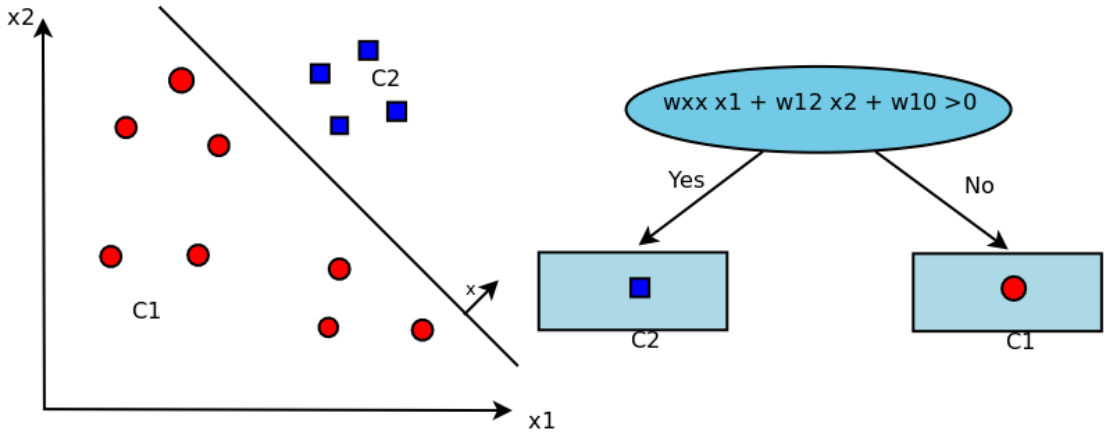
Univariate Decision Tree

- Each internal node, the test uses only one of the input dimensions, e.g, x_j is discrete taking one of n possible values leads to n-way split
- The internal decision node m implements a test function $f_m(x)$ (discriminant in the d-dim input space dividing it into smaller regions) with outcomes labeling the branch; starts at root and ends at leaf
- $f_m(x) : x_j > w_{m0}$; w_{m0} - threshold. Decision node divides input space into: $L_m = \{x|x_j > w_{m0}\}$ and $R_m = \{x|x_j \leq w_{m0}\}$; a binary split orthogonal to each other. The leaf nodes define hyperrectangles in the input space



Multivariate Trees

- In case of univariate tree, only one input dimension is used at a split. In multivariate tree, at a decision node, all input dimensions can be used and thus it is more general, the node becomes more flexible
- A binary linear multivariate node is defined as $f_m(\mathbf{x}) : \mathbf{w}_m^T \mathbf{x} + w_{m0} > 0$; defines a hyperplane with some orientation
- Successive nodes on a path from the root to a leaf further divide these, and leaf nodes define polyhedra in the input space



Multivariate Trees

- Flexibility can be further increased by using a nonlinear multivariate node, e.g., $f_m(\mathbf{x}) : \mathbf{x}^T \mathbf{W}_m \mathbf{x} + \mathbf{w}_m^T \mathbf{x} + w_{m0} > 0$ defines a quadratic node; or $f_m(\mathbf{x}) : ||\mathbf{x} - \mathbf{c}_m|| \leq \alpha_m$, defines a sphere node where c_m is the center and α_m is the radius
- The univariate node is a special case when all but one of w_{mj} are 0. Thus, the previous univariate also defines a linear discriminant but one that is orthogonal to axis x_j, intersecting it at w_{m0} and paraller to all other x_j
- Prove that - In a univariate node there are d possible orientations (w_m) and $N_m - 1$ possible threshold $(-w_{m0})$, making an exhaustive search possible. In a multivariate node, there are $2^d \begin{pmatrix} N_m \\ d \end{pmatrix}$ possible hyperplanes

Multivariate Trees

- When we use univariate nodes, our approx uses piecewise, axis aligned hyperplanes
- With linear multivariate nodes, we can use arbitrary hyperplanes and do a better approx using fewer nodes
- If the underlying discriminant function is curved, nonlinear works better
- The branching factor has a similar effect - it specifies the no of discriminants that a node defines: binary defines one disciminant separating the input space into two. An n-way node separates into n
- There is a dependency among the complexity of a node, the branching factor, and tree size
- With simple nodes and low branching factors, one may grow large trees, but such trees, for e.g., with univariate binary nodes are more interpretable
- Linear multivariate nodes are more difficult to interpret.
- More complex nodes also require more data and are prone to overfitting as we get down the tree and have less and less data
- If the nodes are complex and the tree is small, we also lose the mainidea of the tree, which is dividing the problem into a set of simple problems
- We can have a very complex classifier in the root that separates all classes from each other, but then there is no tree!

Omnivariate Decision Trees

- Is a hybrid tree architecture where the tree may have univariate, linear multivariate, or nonlinear multivariate nodes
- Idea is, at each decision node, which corresponds to a different subproblem defined by the subset of the training data reaching that node, a different model may be appropriate
- At each node, candidate nodes of different types are trained and compared using a statistical test on a validation set to determine which one generalizes the best
- The simpler one is chosen unless a more complex one is shown to have significantly higher accuracy
- More complex nodes are used early in the tree, closer to the root, and as we go down the tree, simple univariate nodes suffice
- As we get closer to leaves, we have simpler problems and less data, so complex nodes would overfit and are rejected by statistical tests
- The no of nodes increases exponentially as we go down the tree, therefore, a large majority of the nodes are univariate and the overall complexity does not increase much
- DT are used more frequently for classification than for regression. They are very popular. They learn and respond quickly, and are accurate in many domains (Murthy 1998). It is interpretable, as can also be written down as a set of IF-THEN rules, which can be validated by human experts with knowledge of application domains

How to Construct Optimal Tree?

- There are exponentially many decision trees that can be constructed from a given set of features
- While some will be more accurate than others, finding the optimal is not computationally feasible
- Nevertheless, a number of efficient algorithms are there to create or grow a reasonably accurate, albeit suboptimal DT, in a reasonable amount of time
- They usually employ a greedy startegy that grows the tree using the most informative attribute at each step and does not allow backtracking
- Most informative attribute splits the set arriving at the node into the most homogeneous subsets

How to Construct Optimal Tree?

- The goal in DT learning
  1. to figure out what questions to ask;
  2. in what order to ask them;
  3. and what answer to predict once you have asked enough questions
- We refer to the questions that you can ask as features. The response to these questions as feature values
- There are a lot of logically possible trees, it is computationally infeasible to consider all of these to try to choose the best one
- Instead, we build out DT greedily - we begin by asking: If I could only ask one question, what question would I ask?

Information

Information is the reduction of uncertainity, and informative attribute will be the ones that result in the largest reduction of uncertainity

Information Content

$I(E) = \log \frac{1}{P(E)} = -\log P(E)$ where $P(E)$ is the prior probability of occurence of a message

Example

If $P_1(E) = P_2(E) = \frac{1}{2}$ then $I(E_1) = I(E_2) = -\log_2(\frac{1}{2}) = 1$ bit
while $P_1(E) = \frac{1}{4}$; $P_2(E) = \frac{3}{4}$ then
$I(E_1) = -\log_2(\frac{1}{4}) = 2 > I(E_2) = -\log_2(\frac{3}{4}) = 0.415$

Messages with high probability of occurence carry least information, messages least expected carry most information

Entropy

Entropy is a measure of the disorder or unpredictability in a system (used for discrete, whereas variance metric is for continuous variable). Given a binary (two-class) classification, C, and a set of examples, S, the class distribution at any node can be written as (p_0, p_1); $p_1 = 1 - p_0$, entropy (H) of S is the sum of the information

Binary classification

$H(S) = -p_0 \log_2 p_0 - p_1 \log_2 p_1$
Maximum value of H by setting $\frac{\partial H}{\partial p} = 0$, occurs when $p = \frac{1}{2}$
$H_{max}(S) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$

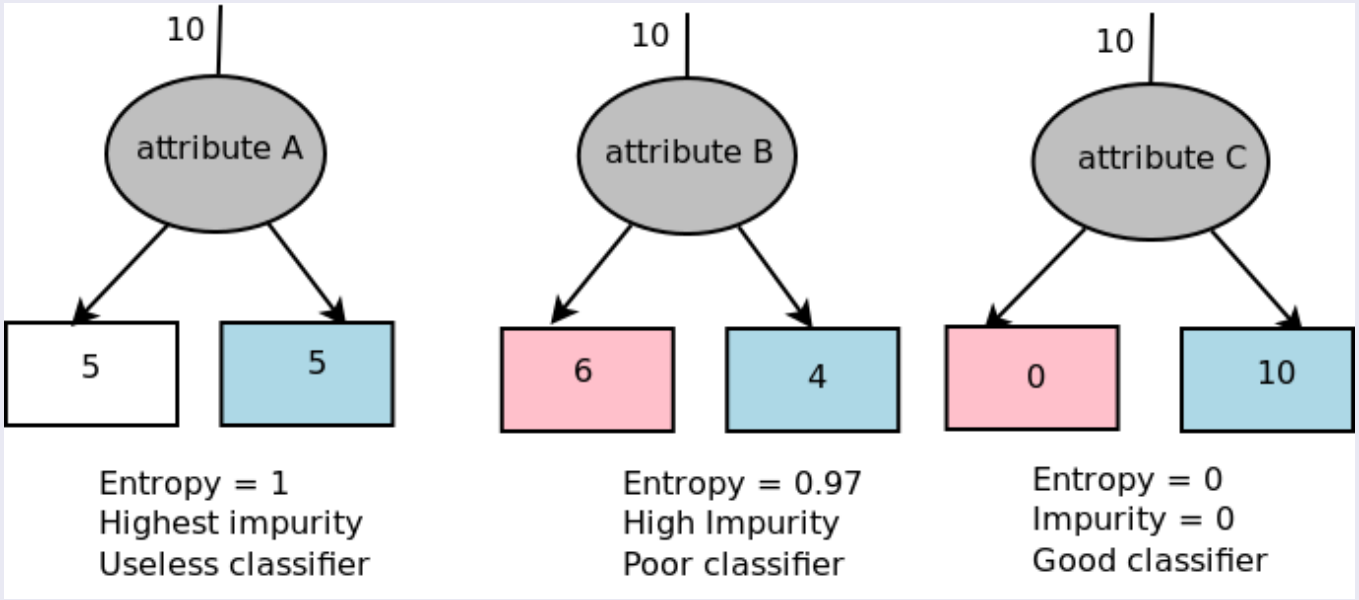
General classification

$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$ where p_i is the proportion of S belonging to class i. For n values H_{max} will occur when all events are equally likely or $p_i = \frac{1}{n}$
$H_{max}(S) = -\sum \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n$

Usefullness of a node

Node

Entropy describes the amount of impurity in a set of features at a node



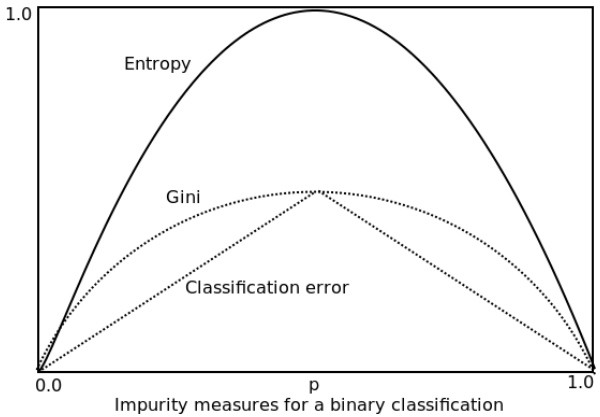
The smaller the degree of impurity/entropy (0), more skewed the class distribution $(0, 1)$, more useful the node. Uniform class distribution $(0.5, 0.5)$, highest entropy (1), useless node.

Other Impurity Measures

Gini impurity and classification error

$Gini(p) = 1 - \sum_i p_i^2$,
classification error $(p) = 1 - max(p_i)$

Gini impurity is the expected error rate if the class label is selected randomly from the class distribution



Impurity measures for a binary classification

All three measures attain a maximum value for a uniform distribution ($p = 0.5$), and a minimum for skewed distribution ($p = 0/1$). All three measures are similar, but Entropy and Gini are differentiable, and hence more subject to numerical optimization (optimal decision). Entropy and Gini are more sensitive to changes in the node probabilities than the misclassification error rate.

Impurity Measures

- Gini measurement is the probability of a random sample being classified correctly if we randomly pick a label according to th distribution in a branch.
- Entropy is a measurement of information (or rather lack thereof). You calculate the information gain by making a split. Which is the difference in entripies. This measures how you reduce the uncertainty about the label.
- Gini is intended for continuous attributes, and Entropy for attributes that occur in classes (e.g. colors)
- Gini will tend to find the largest class, and entropy tends to find groups of classes that make up 50%
- Gini to minimize misclassification Entropy for exploratory analysis. Some studies show this doesnt matter these differ less than 2

Information Gain

Information Gain is used to choose the best attribute for each node

Information Gain

$Gain(S, A) = Impurity(S) - \sum_i^k \frac{|S_{vi}|}{|S|} Impurity(Svi)$ where attribute A has a set of values $\{v_1, v_2 \cdots v_k\}$, and the number of examples within S with the values v_i is $|S_{vi}|$.

The first term is just the impurity of the original collection S
The second term is the expected value of the impurity after S is partitioned using attribute A (the sum of the impurities of each subset S_{vi}, weighted by the fraction of examples that belong to S_{vi}).

Example

W	FV	MC	D
S	Y	R	TV
S	N	R	PO
C	Y	R	TV
R	Y	P	TV
R	N	R	SI
R	Y	P	TV
C	N	P	TV
C	N	R	SI
C	Y	R	TV
S	N	R	PO

Entropy

$S = -0.6 \log_2 0.6 - 0.2 \log_2 0.2 - 2 \times (0.1 \log_2 0.1) = 1.571$

Example

W	FV	MC	D
S	Y	R	TV
S	N	R	PO
C	Y	R	TV
R	Y	P	TV
R	N	R	SI
R	Y	P	TV
C	N	P	TV
C	N	R	SI
C	Y	R	TV
S	N	R	PO

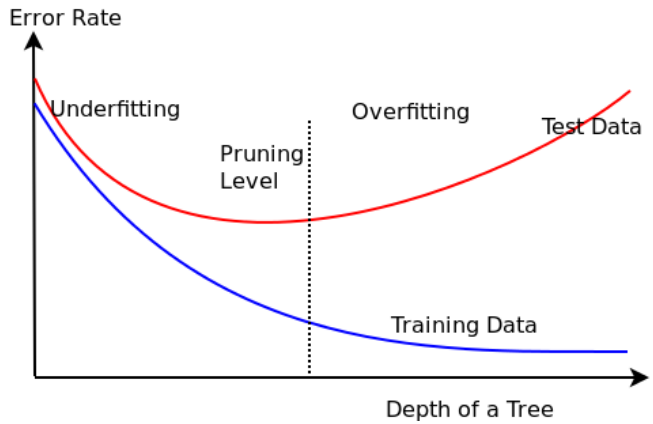
Entropy

$S = -0.6 \log_2 0.6 - 0.2 \log_2 0.2 - 2 \times (0.1 \log_2 0.1) = 1.571$

Information Gain

$Gain(S, F) = 1.571 - (\frac{|S_y|}{10}) \times Entropy(S_y) - (\frac{|S_n|}{10}) \times Entropy(S_n)$
$= 1.571 - (0.5) \times 0 - (0.5) \times (-0.4 \log_2 0.4 - 3 \times (0.2 \log_2 0.2) = 1.922) = 0.61$
$Gain(S, W) =$
$1.571 - (\frac{|S_s|}{10}) \times Entropy(S_s) - (\frac{|S_c|}{10}) \times Entropy(S_c) - (\frac{|S_r|}{10}) \times Entropy(S_r)$
$= 1.571 - (0.3) \times (0.918) - (0.4) \times (0.8113) - (0.3) \times (0.918) = 0.71$
$Gain(S, M) = 1.571 - (0.7) \times (1.842) - (0.3) \times 0 = 0.2816$
Weather should be the first(root)node .Similary, we look at the branches.
For S_{sunny} branch, classes (TV, PO, SI) are not same, we need another decision node here. Same situation occurs for S_{cold} and S_{rainy}.
For sunny: $H(S) = -0.1 \log_2 0.1 - 0.2 \log_2 0.2 = 0.981$
$Gain(S_{sunny}, F) = 0.981 - (\frac{|S_y|}{|S|}) \times Entropy(S_y) - (\frac{|S_n|}{|S|}) \times Entropy(S_n)$
$= 0.918 - (1/3) \times 0 - (2/3) \times 0 = 0.918$
$Gain(S_{sunny}, M) = 0.981 - (\frac{|S_r|}{|S|}) \times Entropy(S_r) - (\frac{|S_p|}{|S|}) \times Entropy(S_p)$
$= 0.918 - (3/3) \times 0.918 - (0/3) \times 0 = 0$
The friend attribute is chosen next. The rest of the tree is left as an exercise.

Decision Tree Issues



- Overfit - each terminal (leaf) node will correspond to a single training point and the full tree is merely a look-up table, we can not expect to generalize to noisy data.
- Underfit - if splitting is stopped too early, then training and test error rates are large (tree is very small), it has yet to learn
- Decide using validation or cross validation until the error is minimized
- Stop when the reduction in impurity falls below a threshold or a node represents less than a certain no of points (5% of total)

Decision Tree Issues

- Prune (large grown trees) - by trimming in a bottom - up fashion all pair of neighboring leaf nodes whose elimination results in small increase in impurity are eliminated, with the common parent becoming parent node. Pruning is incorporated in C4.5 successor to ID3.
- Tend to give rectangular regions, unless more computationally expensive strategies are used. Oblique trees can be grown using test conditions as $w_{11}x_1 + w_{12}x_2 + w_{10} > 0$
- Missing attribute values are estimated based
  1. on other examples for which this attribute has a known value or
  2. by the value which is most common among training samples at this node in the same classification
  3. to assign a probability to each of the possible values of A. A fraction, $p(x_v)$ of x is now distributed down the branch for A when $x = v$ and the rest down the other branches similarly

Decision Tree Issues

- When comparing binary splits with multiway splits, the multiway split is inherently favored.
- To avoid this, the gain should be normalized by dividing by $\sum p_i \log_2 p_i$ (summed over the number of splits, k)
- If each attribute has the same no of cases, then $p_i = \frac{1}{k}$ and the normalizing factor is $\log_2 k$
- With multiway splits, it is important to ensure that there are sufficient no of cases associated with each choice to make reliable predictions

Information Gain Ratio (Normalized Information Gain $C4.5$)

- $GainRatio(T, A) \equiv \frac{Gain(T,A)}{SplitInformation(T,A)}$, is the gain in entropy due to splitting using attribute A
- The sum of the entropy of the nodes after splitting on attribute A is $SplitInformation(T, A) \equiv -\sum_{i=1}^{c} \frac{|T_i|}{|T|} \log_2 \frac{|T_i|}{|T|}$, where T_i is the subset of T for which A has value v_i. $SplitInformation(T, A) = \log_2 n$ if there are n attributes. If the attribute is having binary output and divides the set exactly in half the value of $SplitInformation(T, A) = 1$. Use attribute with highest IGR for splitting
- If there are attributes with differing costs, use DTs that use low-cost attributes when possible and high-cost attributes only when needed. Use $\frac{Gain(T,A)}{Cost(A)}$ or $\frac{Gain^2(T,A)}{Cost(A)}$ or $\frac{2^{Gain(T,A)}-1}{(Cost(A)+1)^w}$ where $w \in \{0, 1\}$ in various situations to account for varying cost of attributes

Algorithms

ID3

The ID3 alg builds a DT based on the largest *IG*.

- Select an attribute for the root node to ensure the maximum *IG*
- Create branches for subranges of attribute values
- Split attribute values according to branches
- Recurse the overall till you get a pure branch

C4.5

The ID3 alg builds a DT based on the largest *IGR*.

- Select an attribute for the root node to ensure the maximum *IGR*
- Create branches for subranges of attribute values
- Split attribute values according to branches
- Recurse the overall till you get a pure branch

One Splitting Rule for Continous Attributes

- Assume we are allowed to create only binary splits for continuous attributes.
- Let the variance of the target variable at the root node be $Var(T)$.
- Let the variance of the target variable at the proposed left and right subtrees be $Var(T_l)$ and $Var(T_r)$.
- Let the proportion of samples in the left and right subtrees be $p(T_l)$ and $p(T_r)$
- One splitting rule can be: Choose the split that makes the reduction in variance the most, i.e., makes the value of $Var(T) - p(T_l)\ Var(T_l) - p(T_r)\ Var(T_r)$ the highest.

Decision Tree Strength and Weakness

Strength

- Easy to understand. Do not require much computation
- Clear indication of importance of attributes (fields/features)
- Most useful for categorical data (adaptive for continuous data)
- It is invariant under scaling and various other transformations
- Is robust to inclusion of irrelevant features, and produces inspectable models

Weakness

- Prone to errors for large number of classes with relatively small number of training examples
- Unstable learners meaning that their hypotheses drastically change when small changes to the training data are made.
- Can be computationally expensive to train. Pruning is also computationally expensive

Inductive bias in DT learning

- ID3 selects in favor of shorter trees over longer ones, and selects trees that place the attributes with higest IG closest to the root
- Bagged decision trees (BDT) work well and obtain solid predictive power
- Random forests takes the diversity one extra level by incorporating the random subspace method: which is only difference between RF and BDT. At each split made by each tree, we consider only a subset of features to make the optimal split on

Tree based models vs linear models?

- If the relationship between dependent and independent variable is well approximated by a linear model, linear regression will outperform tree based model.
- If there is a high non-linearity and complex relationship between dependent and independent variables, a tree model will outperform a classical regression method.
- If you need to build a model which is easy to explain to people, a decision tree model will always do better than a linear model. Decision tree models are even simpler to interpret than linear regression!

Tree based models vs instance based models vs statistical models

- Each leaf node corresponds to a "bin" except that the bins need not be the same size (as in Parzen window) or contain an equal no of training instances (as in K-nearest neighbor)
- The bin divisions are not done based only on similarity in the input space, but supervised output information through entropy or mean square error is also used
- In DT, the leaf ("bin") is found much faster with smaller number of comparisons
- Once constructed the DT, does not store all the training set but only the structure of the tree, the parameters of the decision nodes, and the output values in leaves; this implies that the space complexity is also much less, as opposed to instance-based non parametric methods that store all the training examples
- Trees code directly the discriminants separating class instances without caring much for how these instances are distributed in the regions
- DT is discriminant-based, whereas the statistical methods are likelihood based in that they explicitly estimate $p(x|C_i)$ before using Bayes' rule and calculating the discriminant
- It directly estimate the discriminant, bypassing the estimatation of class densities

Bagging

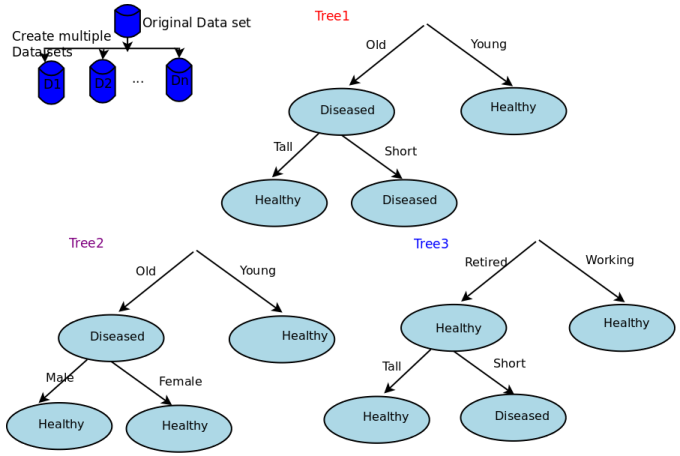
- Input: n labeled training examples $(x_i, y_i), i = 1, \cdots, n$. Repeat k times: Select m samples out of n with replacement to get training set S_i
- Train classifier (decision tree, kNN, perceptron, etc) h_i on S_i
- Output: Classifiers $h_1, \cdots, h_k$
- Classification: On test example x, output majority $(h_1, \cdots, h_k)$

Bagging

- Popular choice: $m = n$
- Still different from working with entire training set. Why?
- $P(S_i = S) = \frac{n!}{n^n}$ (tiny number, exponentially small in n)
- $P((x_i, y_i) \text{ not in } S_i) = (1 - \frac{1}{n})^n \approx \exp^{-1}$
- For large databases, about 37% of the data is left out
- You get k different models

Bagged decision trees (BDT)

- With three different sets of Attributes and databases, learn multiple decision trees



- New Sample: old, retired, male, short
- Tree Predictions: diseased, healthy, diseased

Random Decision Forest

- RDF is an extension of decision trees via bagging
- Draw a bootstrap of size N from the training data
- Grow a DT, T_i for this bootstrap
  1. Select $m < M$ features at random from the entire set of features (attributes), M total features, the value of m is held constant while we grow the forest
  2. Select the best split point among the m attributes
  3. Split the node into daughter nodes
  4. Each tree is grown to the largest extent possible and there is no pruning
- Repeat the previous step for all the bootstraps
- Output the ensemble of trees
- Classify or predict new data by aggregating the predictions of the ntree ensemble trees (i.e., majority votes for classification, average for regression)
- RDF are currently used by Microsoft in Bing Ads, Microsoft hand pose estimation in Kinect, and Image segmentation

Random Decision Forest

Advantages

- Can model non-linear decision boundaries also, can solve both type of problems i.e. classification and regression and does a decent estimation at both fronts.
- Works on both continous and categorical inputs
- Good classification performance
- Reduces the variance of the classifier
- The power to handle large data set with higher dimensionality. It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods. Further, the model outputs Importance of variable, which can be a very handy feature (on some random data set)
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing errors in data sets where classes are imbalanced.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.

Random Decision Forest

Advantages

- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing. It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data. Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection. It offers an experimental method for detecting variable interactions.

Random Decision Forest

Disadvantages

- Training is comparatively slow. It is difficult to analyze the obtained decision rules
- Good at classification but not as good as for regression problem as it does not give precise continuous nature predictions. In case of regression, it doesnt predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.
- RF can feel like a black box approach for statistical modelers - very little control on what the model does. You can at best - try different parameters and random seeds!

Forest Error Rate depends on

- The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.
- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.
- Reducing m reduces both the correlation and the strength. Increasing it increases both. Somewhere in between is an "optimal" range of m - usually quite wide.

Out-of-bag (oob) error, proximities and variable importance

- Random Forest involves sampling of the input data with replacement called as bootstrap sampling. Here one third of the data is not used for training and can be used to testing. These are called the out of bag samples, used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance. Error estimated on these out of bag samples is known as out of bag error. Study of error estimates by Out of bag, gives evidence to show that the out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need for a set aside test set.
- Put each case left out in the construction of the k^{th} tree down the k^{th} tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.
- After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.
- In every tree grown in the forest, put down the oob cases and count the number of votes cast for the correct class. Now randomly permute the values of variable m in the oob cases and put these cases down the tree. Subtract the number of votes for the correct class in the variable-m-permuted oob data from the number of votes for the correct class in the untouched oob data. The average of this number over all trees in the forest is the raw importance score for variable m.

Regression Trees

Classification vs Regression Trees

- Regression trees are used when dependent variable is continuous. Classification trees are used when dependent variable is categorical.
- In case of regression tree, the value obtained by terminal nodes in the training data is the mean response of observation falling in that region. Thus, if an unseen data observation falls in that region, make its prediction with mean value.
- In case of classification tree, the value (class) obtained by terminal node in the training data is the mode of observations falling in that region. Thus, if an unseen data observation falls in that region, make its prediction with mode value.
- Both the trees divide the predictor space (independent variables) into distinct and non-overlapping regions. For the sake of simplicity, you can think of these regions as high dimensional boxes or boxes.
- Both the trees follow a top-down greedy approach known as recursive binary splitting. We call it as "top-down" because it begins from the top of tree when all the observations are available in a single region and successively splits the predictor space into two new branches down the tree. It is known as "greedy" because, the algorithm cares (looks for best variable available) about only the current split, and not about future splits which will lead to a better tree.
- This splitting process is continued until a user defined stopping criteria is reached.
- In both the cases, the splitting process results in fully grown trees until the stopping criteria is reached. But, the fully grown tree is likely to overfit data, leading to poor accuracy on unseen data. This bring "pruning". Pruning is one of the technique used to tackle overfitting.