Decision tree learning

From Wikipedia, the free encyclopedia

Decision tree learning uses a decision tree as a predictive model which maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called **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**.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making). This page deals with decision trees in data mining.

Contents

General

Decision tree learning is a method commonly used in data mining.[1] The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown in the diagram at right. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

A decision tree is a simple representation for classifying examples. For this section, assume that all of the input features have finite discrete domains, and there is a single target feature called the classification. Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target or output feature or the arc leads to a subordinate decision node on a different input feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. See the examples illustrated in the figure for spaces that have and have not been partitioned using recursive partitioning, or recursive binary

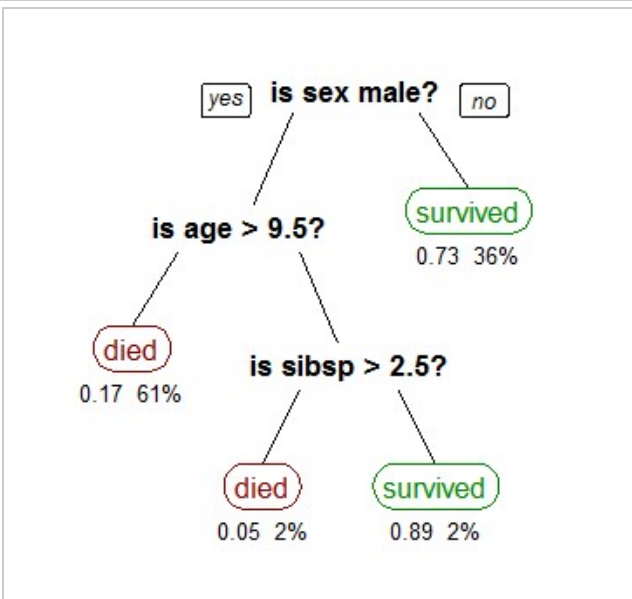
splitting. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of *top-down induction of decision trees* (TDIDT) [2] is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.

In data mining, decision trees can be described also as the combination of mathematical and computational techniques to aid the description, categorization and generalization of a given set of data.

Data comes in records of the form:

$$(\mathbf{x}, Y) = (x_1, x_2, x_3, \ldots, x_k, Y)$$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector **x** is composed of the input variables, x_1, x_2, x_3 etc., that are used for that task.

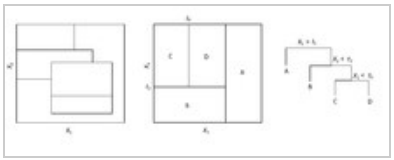


A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of outcome and the percentage of observations in the leaf.

Types

Decision trees used in data mining are of two main types:

- **Classification tree** analysis is when the predicted outcome is the class to which the data belongs.
- **Regression tree** analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).



Left: A partitioned two-dimensional feature space. These partitions could not have resulted from recursive binary splitting. Middle: A partitioned two-dimensional feature space with partitions that did result from recursive binary splitting. Right: A tree corresponding to the partitioned feature space in the middle. Notice the convention that when the expression at the split is true, the tree follows the left branch. When the expression is false, the right branch is followed.

The term **Classification And Regression Tree (CART)** analysis is an umbrella term used to refer to both of the above procedures, first introduced by Breiman et al.[3] Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split.[3]

Some techniques, often called *ensemble* methods, construct more than one decision tree:

- **Bagging** decision trees, an early ensemble method, builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction.[4]
- A **Random Forest** classifier uses a number of decision trees in order to improve the classification rate.
- **Boosted Trees** can be used for regression-type and classification-type problems.[5][6]
- **Rotation forest** - in which every decision tree is trained by first applying principal component analysis (PCA) on a random subset of the input features.[7]

A special case of a decision tree is a Decision list,[8] which is a one-sided decision tree, so that every internal node has exactly 1 leaf node and exactly 1 internal node as a child (except for the bottommost node, whose only child is a single leaf node). While less expressive, decision lists are arguably easier to understand than general decision trees due to their added sparsity, permit non-greedy learning methods[9] and monotonic constraints to be imposed.[10]

Decision tree learning is the construction of a decision tree from class-labeled training tuples. A decision tree is a flow-chart-like structure, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf (or terminal) node holds a class label. The topmost node in a tree is the root node.

There are many specific decision-tree algorithms. Notable ones include:

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.[11]
- MARS: extends decision trees to handle numerical data better.
- Conditional Inference Trees. Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning.[12][13]

ID3 and CART were invented independently at around the same time (between 1970 and 1980), yet follow a similar approach for learning decision tree from training tuples.

Metrics

Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items.[14] Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets. Some examples are given below. These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

Gini impurity

Used by the CART (classification and regression tree) algorithm, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Gini impurity can be computed by summing the probability f_i of an item with label i being chosen times the probability $1 - f_i$ of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

To compute Gini impurity for a set of items with J classes, suppose $i \in \{1, 2, \ldots, J\}$, and let f_i be the fraction of items labeled with class i in the set.

$$I_G(f) = \sum_{i=1}^{J} f_i(1 - f_i) = \sum_{i=1}^{J} (f_i - {f_i}^2) = \sum_{i=1}^{J} f_i - \sum_{i=1}^{J} {f_i}^2 = 1 - \sum_{i=1}^{J} {f_i}^2 = \sum_{i \neq k} f_i f_k$$

Information gain

Used by the ID3, C4.5 and C5.0 tree-generation algorithms. Information gain is based on the concept of entropy from information theory.

Entropy is defined as below

$$H(T) = I_E(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

where $p_1, p_2, \ldots$ are fractions that add up to 1 and represent the percentage of each class present in the child node that results from a split in the tree.[15]

Information Gain = Entropy(parent) - Weighted Sum of Entropy(Children)

$$IG(T,a) = H(T) - H(T|a)$$

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information which is measured in bits, not to be confused with the unit of computer memory. For each node of the tree, the information value "represents the expected amount of information that would be needed to specify whether a new instance should be classified yes or no, given that the example reached that node".[15]

Consider an example data set with four attributes: outlook (sunny, overcast, rainy), temperature (hot, mild, cool), humidity (high, normal), and windy (true, false), with a binary (yes or no) target variable, play, and 14 data points. To construct a decision tree on this data, we need to compare the information gain of each of four trees, each split on one of the four features. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.

The split using the feature windy results in two children nodes, one for a windy value of true and one for a windy value of false. In this data set, there are six data points with a true windy value, three of which have a play value of yes and three with a play value of no. The eight remaining data points with a windy value of false contain two no's and six yes's. The information of the windy=true node is calculated using the entropy equation above. Since there is an equal number of yes's and no's in this node, we have

$$I_E([3,3]) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$$

For the node where windy=false there were eight data points, six yes's and two no's. Thus we have

$$I_E([6,2]) = -(6/8)\log_2(6/8) - (2/8)\log_2(2/8) = -(3/4)\log_2(3/4) - (1/4)\log_2(1/4) = 0.8112781$$

To find the information of the split, we take the weighted average of these two numbers based on how many observations fell into which node.

$$I_E([3,3],[6,2]) = I_E(windyornot) = (6/14)(1) + (8/14)(0.8112781) = 0.8921589$$

To find the information gain of the split using windy, we must first calculate the information in the data before the split. The original data contained nine yes's and five no's.

$$I_E([9,5]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940286$$

Now we can calculate the information gain achieved by splitting on the windy feature.

$$IG(windy) = I_E([9,5]) - I_E([3,3],[6,2]) = 0.940286 - 0.8921589 = 0.0481271$$

To build the tree, the information gain of each possible first split would need to be calculated. The best first split is the one that provides the most information gain. This process is repeated for each impure node until the tree is complete. This example is adapted from the example appearing in Witten et al.[15]

Variance reduction

Introduced in CART,[3] variance reduction is often employed in cases where the target variable is continuous (regression tree), meaning that use of many other metrics would first require discretization before being applied. The variance reduction of a node N is defined as the total reduction of the variance of the target variable x due to the split at this node:

$$I_V(N) = \frac{1}{|S|^2}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|^2}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|^2}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i - x_j)^2\right)$$

where , S_t, and S_f are the set of presplit sample indices, set of sample indices for which the split test is true, and set of sample indices for which the split test is false, respectively. Each of the above summands are indeed variance estimates, though, written in a form without directly referring to the mean.

Decision tree advantages

Amongst other data mining methods, decision trees have various advantages:

- **Simple to understand and interpret.** People are able to understand decision tree models after a brief explanation. Trees can also be displayed graphically in a way that is easy for non-experts to interpret.[16]
- **Able to handle both numerical and categorical data.**[16] Other techniques are usually specialised in analysing datasets that have only one type of variable. (For example, relation rules can be used only with nominal variables while neural networks can be used only with numerical variables.)
- **Requires little data preparation.** Other techniques often require data normalization. Since trees can handle qualitative predictors, there is no need to create dummy variables.[16]
- **Uses a white box model.** If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model, the explanation for the results is typically difficult to understand, for example with an artificial neural network.
- **Possible to validate a model using statistical tests.** That makes it possible to account for the reliability of the model.
- **Robust.** Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.
- **Performs well with large datasets.** Large amounts of data can be analysed using standard computing resources in reasonable time.
- **Mirrors human decision making more closely than other approaches.**[16] This could be useful when modeling human decisions/behavior.

Limitations

- Trees do not tend to be as accurate as other approaches.[16]
- Trees can be very non-robust. A small change in the training data can result in a big change in the tree, and thus a big change in final predictions.[16]
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts.[17][18] Consequently, practical decision-tree learning algorithms are based on heuristics such as the greedy algorithm where locally-optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally-optimal decision tree. To reduce the greedy effect of local-optimality some methods such as the dual information distance (DID) tree were proposed.[19] [1] (http://www.eng.tau.ac.il/~bengal/DID.pdf)

- Decision-tree learners can create over-complex trees that do not generalize well from the training data. (This is known as overfitting.[20]) Mechanisms such as pruning are necessary to avoid this problem (with the exception of some algorithms such as the Conditional Inference approach, that does not require pruning [12][13]).
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems. In such cases, the decision tree becomes prohibitively large. Approaches to solve the problem involve either changing the representation of the problem domain (known as propositionalization)[21] or using learning algorithms based on more expressive representations (such as statistical relational learning or inductive logic programming).
- For data including categorical variables with different numbers of levels, information gain in decision trees is biased in favor of those attributes with more levels.[22] However, the issue of biased predictor selection is avoided by the Conditional Inference approach.[12]

Extensions

Decision graphs

In a decision tree, all paths from the root node to the leaf node proceed by way of conjunction, or *AND*. In a decision graph, it is possible to use disjunctions (ORs) to join two more paths together using Minimum message length (MML).[23] Decision graphs have been further extended to allow for previously unstated new attributes to be

learnt dynamically and used at different places within the graph.[24] The more general coding scheme results in better predictive accuracy and log-loss probabilistic scoring. In general, decision graphs infer models with fewer leaves than decision trees.

Alternative search methods

Evolutionary algorithms have been used to avoid local optimal decisions and search the decision tree space with little *a priori* bias.[25][26]

It is also possible for a tree to be sampled using MCMC.[27]

The tree can be searched for in a bottom-up fashion.[28]

See also

- Decision tree pruning
- Binary decision diagram
- CHAID
- CART
- ID3 algorithm
- C4.5 algorithm
- Decision stump
- Decision list
- Incremental decision tree
- Alternating decision tree
- Structured data analysis (statistics)
- Logistic model tree
- Hierarchical clustering

Implementations

Many data mining software packages provide implementations of one or more decision tree algorithms. Several examples include Salford Systems CART (which licensed the proprietary code of the original CART authors[3]), IBM SPSS Modeler, RapidMiner, SAS Enterprise Miner, Matlab, R (an open source software environment for statistical computing which includes several CART implementations such as rpart, party and randomForest packages), Weka (a free and open-source data mining suite, contains many decision tree algorithms), Orange (a free data mining software suite, which includes the tree module orngTree (http://www.ailab.si/orange/doc/modules/orngTree.htm)), KNIME, Microsoft SQL Server [2] (http://technet.microsoft.com/en-us/library/cc645868.aspx), and scikit-learn (a free and open-source machine learning library for the Python programming language).

References

1. Rokach, Lior; Maimon, O. (2008). *Data mining with decision trees: theory and applications*. World Scientific Pub Co Inc. ISBN 978-9812771711.
2. Quinlan, J. R., (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers
3. Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. ISBN 978-0-412-04841-8.
4. Breiman, L. (1996). Bagging Predictors. "Machine Learning, 24": pp. 123-140.
5. Friedman, J. H. (1999). *Stochastic gradient boosting.* Stanford University.
6. Hastie, T., Tibshirani, R., Friedman, J. H. (2001). *The elements of statistical learning : Data mining, inference, and prediction.* New York: Springer Verlag.
7. Rodriguez, J.J. and Kuncheva, L.I. and Alonso, C.J. (2006), Rotation forest: A new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(10):1619-1630.
8. Rivest, Ron (Nov 1987). "Learning Decision Lists" (PDF). *Machine Learning*. **3** (2): 229–246. doi:10.1023/A:1022607331053.

9. Letham, Ben; Rudin, Cynthia; McCormick, Tyler; Madigan, David (2015). "Interpretable Classifiers Using Rules And Bayesian Analysis: Building A Better Stroke Prediction Model". *Annals of Applied Statistics*. **9**: 1350–1371. arXiv:1511.01644. doi:10.1214/15-AOAS848.
10. Wang, Fulton; Rudin, Cynthia (2015). "Falling Rule Lists" (PDF). *Journal of Machine Learning Research*. **38**.
11. Kass, G. V. (1980). "An exploratory technique for investigating large quantities of categorical data". *Applied Statistics*. **29** (2): 119–127. doi:10.2307/2986296. JSTOR 2986296.
12. Hothorn, T.; Hornik, K.; Zeileis, A. (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework". *Journal of Computational and Graphical Statistics*. **15** (3): 651–674. doi:10.1198/106186006X133933. JSTOR 27594202.
13. Strobl, C.; Malley, J.; Tutz, G. (2009). "An Introduction to Recursive Partitioning: Rationale, Application and Characteristics of Classification and Regression Trees, Bagging and Random Forests". *Psychological Methods*. **14** (4): 323–348. doi:10.1037/a0016973.
14. Rokach, L.; Maimon, O. (2005). "Top-down induction of decision trees classifiers-a survey". *IEEE Transactions on Systems, Man, and Cybernetics, Part C*. **35** (4): 476–487. doi:10.1109/TSMCC.2004.843247.
15. Witten, Ian; Frank, Eibe; Hall, Mark (2011). *Data Mining*. Burlington, MA: Morgan Kaufmann. pp. 102–103. ISBN 978-0-12-374856-0.
16. Gareth, James; Witten, Daniela; Hastie, Trevor; Tibshirani, Robert (2015). *An Introduction to Statistical Learning*. New York: Springer. p. 315. ISBN 978-1-4614-7137-0.
17. Hyafil, Laurent; Rivest, RL (1976). "Constructing Optimal Binary Decision Trees is NP-complete". *Information Processing Letters*. **5** (1): 15–17. doi:10.1016/0020-0190(76)90095-8.
18. Murthy S. (1998). Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*
19. Ben-Gal I. Dana A., Shkolnik N. and Singer (20). "Efficient Construction of Decision Trees by the Dual Information Distance Method" (PDF). Quality Technology & Quantitative Management (QTM), 11(1), 133-147. Check date values in: `|date=` (help)
20. "Principles of Data Mining". 2007. doi:10.1007/978-1-84628-766-4. ISBN 978-1-84628-765-7.
21. Horváth, Tamás; Yamamoto, Akihiro, eds. (2003). "Inductive Logic Programming". Lecture Notes in Computer Science. **2835**. doi:10.1007/b13700. ISBN 978-3-540-20144-1.
22. Deng,H.; Runger, G.; Tuv, E. (2011). *Bias of importance measures for multi-valued attributes and solutions*. Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN). pp. 293–300.
23. http://citeseer.ist.psu.edu/oliver93decision.html
24. Tan & Dowe (2003) (http://www.csse.monash.edu.au/~dld/Publications/2003/Tan+Dowe2003_MMLDecisionGraphs.pdf)
25. Papagelis A., Kalles D.(2001). Breeding Decision Trees Using Evolutionary Techniques, Proceedings of the Eighteenth International Conference on Machine Learning, p.393-400, June 28-July 01, 2001
26. Barros, Rodrigo C., Basgalupp, M. P., Carvalho, A. C. P. L. F., Freitas, Alex A. (2011). A Survey of Evolutionary Algorithms for Decision-Tree Induction (http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5928432). IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, vol. 42, n. 3, p. 291-312, May 2012.
27. Chipman, Hugh A., Edward I. George, and Robert E. McCulloch. "Bayesian CART model search." Journal of the American Statistical Association 93.443 (1998): 935-948.
28. Barros R. C., Cerri R., Jaskowiak P. A., Carvalho, A. C. P. L. F., A bottom-up oblique decision tree induction algorithm (http://dx.doi.org/10.1109/ISDA.2011.6121697). Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011).

External links

- Building Decision Trees in Python (http://onlamp.com/lpt/a/6464) From O'Reilly.
- An Addendum to "Building Decision Trees in Python" (http://www.oreillynet.com/mac/blog/2007/06/an_addendum_to_building_decisi.html) From O'Reilly.
- Decision Trees Tutorial (http://people.revoledu.com/kardi/tutorial/DecisionTree/index.html) using Microsoft Excel.
- Decision Trees page at aitopics.org (http://aitopics.org/topic/decision-tree-learning), a page with commented links.
- Decision tree implementation in Ruby (AI4R) (http://ai4r.org/index.html)
- Evolutionary Learning of Decision Trees in C++ (http://www.cs.uwaterloo.ca/~mgrzes/code/evoldectrees/)
- Java implementation of Decision Trees based on Information Gain (https://github.com/kobaj/JavaDecisionTree)
- A very explicit explanation of information gain as splitting criterion (http://christianherta.de/lehre/dataScience/machineLearning/decision-trees.html)

Categories: Decision trees | Classification algorithms