Published in final edited form as:
Nat Biotechnol. 2008 September ; 26(9): 1011–1013. doi:10.1038/nbt0908-1011.

What are decision trees?

Carl Kingsford and **Steven L Salzberg**
Department of Computer Science, Institute for Advanced Computer Studies and Center for Bioinformatics and Computational Biology, University of Maryland, College Park, Maryland 20742, USA. e-mail: carlk@cs.umd.edu or salzberg@umiacs.umd.edu

Abstract

Decision trees have been applied to problems such as assigning protein function and predicting splice sites. How do these classifiers work, what types of problems can they solve and what are their advantages over alternatives?

Many scientific problems entail labeling data items with one of a given, finite set of classes based on features of the data items. For example, oncologists classify tumors as different known cancer types using biopsies, patient records and other assays. Decision trees, such as C4.5 (ref. 1), CART2 and newer variants, are classifiers that predict class labels for data items. Decision trees are at their heart a fairly simple type of classifier, and this is one of their advantages.

Decision trees are constructed by analyzing a set of training examples for which the class labels are known. They are then applied to classify previously unseen examples. If trained on high-quality data, decision trees can make very accurate predictions3.

Classifying with decision trees

A decision tree classifies data items (Fig. 1a) by posing a series of questions about the features associated with the items. Each question is contained in a node, and every internal node points to one child node for each possible answer to its question. The questions thereby form a hierarchy, encoded as a tree. In the simplest form (Fig. 1b), we ask yes-or-no questions, and each internal node has a 'yes' child and a 'no' child. An item is sorted into a class by following the path from the topmost node, the root, to a node without children, a leaf, according to the answers that apply to the item under consideration. An item is assigned to the class that has been associated with the leaf it reaches. In some variations, each leaf contains a probability distribution over the classes that estimates the conditional probability that an item reaching the leaf belongs to a given class. Nonetheless, estimation of unbiased probabilities can be difficult4.

Questions in the tree can be arbitrarily complicated, as long as the answers can be computed efficiently. A question's answers can be values from a small set, such as {A,C,G,T}. In this case, a node has one child for each possible value. In many instances, data items will have real-valued features. To ask about these, the tree uses yes/no questions of the form "is the value > *k*?" for some threshold *k*, where only values that occur in the data need to be tested as possible thresholds. It is also possible to use more complex questions, taking either linear or logical combinations of many features at once5.

Decision trees are sometimes more interpretable than other classifiers such as neural networks and support vector machines because they combine simple questions about the data in an understandable way. Approaches for extracting decision rules from decision trees have also been successful1. Unfortunately, small changes in input data can sometimes lead to large changes in the constructed tree. Decision trees are flexible enough to handle items with a mixture of real-valued and categorical features, as well as items with some missing features. They are expressive enough to model many partitions of the data that are not as easily achieved with classifiers that rely on a single decision boundary (such as logistic regression or support vector machines). However, even data that can be perfectly divided into classes by a hyperplane may require a large decision tree if only simple threshold tests are used. Decision trees naturally support classification problems with more than two classes and can be modified to handle regression problems. Finally, once constructed, they classify new items quickly.

Constructing decision trees

Decision trees are grown by adding question nodes incrementally, using labeled training examples to guide the choice of questions1,2. Ideally, a single, simple question would perfectly split the training examples into their classes. If no question exists that gives such a perfect separation, we choose a question that separates the examples as cleanly as possible.

A good question will split a collection of items with heterogeneous class labels into subsets with nearly homogeneous labels, stratifying the data so that there is little variance in each stratum. Several measures have been designed to evaluate the degree of inhomogeneity, or impurity, in a set of items. For decision trees, the two most common measures are entropy and the Gini index. Suppose we are trying to classify items into m classes using a set of training items E. Let p_i ($i = 1,\ldots,m$) be the fraction of the items of E that belong to class i. The entropy of the probability distribution $(p_i)_{i=1}^{m}$ gives a reasonable measure of the impurity of the set E. The entropy, $-\sum_{i=1}^{m} p_i \log p_i$, is lowest when a single p_i equals 1 and all others are 0, whereas it is maximized when all the p_i are equal. The Gini index2, another common measure of impurity, is computed by $1 - \sum_{i=1}^{m} p_i^2$. This is again zero when the set E contains items from only one class.

Given a measure of impurity I, we choose a question that minimizes the weighted average of the impurity of the resulting children nodes. That is, if a question with k possible answers divides E into subsets $E_1\ldots,E_k$, we choose a question to minimize $\sum_{j=1}^{k} (|E_j|/|E|) I(E_j)$. In many cases, we can choose the best question by enumerating all possibilities. If I is the entropy function, then the difference between the entropy of the distribution of the classes in the parent node and this weighted average of the children's entropy is called the information gain. The information gain, which is expressible via the Kullback-Leibler divergence6, always has a nonnegative value.

We continue to select questions recursively to split the training items into ever-smaller subsets, resulting in a tree. A crucial aspect to applying decision trees is limiting the complexity of the learned trees so that they do not overfit the training examples. One technique is to stop splitting when no question increases the purity of the subsets more than a small amount. Alternatively, we can choose to build out the tree completely until no leaf can be further subdivided. In this case, to avoid overfitting the training data, we must prune the tree by deleting nodes. This can be done by collapsing internal nodes into leaves if doing so reduces the classification error on a held-out set of training examples1. Other approaches, relying on ideas such as minimum description length1,6,7, remove nodes in an attempt to explicitly balance the complexity of the tree with its fit to the training data. Cross-validation on left-out training examples should be used to ensure that the trees generalize beyond the examples used to construct them.

Ensembles of decision trees and other variants

Although single decision trees can be excellent classifiers, increased accuracy often can be achieved by combining the results of a collection of decision trees8–10. Ensembles of decision trees are sometimes among the best performing types of classifiers3. Random forests and boosting are two strategies for combining decision trees.

In the random forests8 approach, many different decision trees are grown by a randomized tree-building algorithm. The training set is sampled with replacement to produce a modified training set of equal size to the original but with some training items included more than once. In addition, when choosing the question at each node, only a small, random subset of the features is considered. With these two modifications, each run may result in a slightly different tree. The predictions of the resulting ensemble of decision trees are combined by taking the most common prediction. Maintaining a collection of good hypotheses, rather than committing to a single tree, reduces the chance that a new example will be misclassified by being assigned the wrong class by many of the trees.

Boosting10 is a machine-learning method used to combine multiple classifiers into a stronger classifier by repeatedly reweighting training examples to focus on the most problematic. In practice, boosting is often applied to combine decision trees. Alternating decision trees11 are a generalization of decision trees that result from applying a variant of boosting to combine weak classifiers based on decision stumps, which are decision trees that consist of a single question. In alternating decision trees, the levels of the tree alternate between standard question nodes and nodes that contain weights and have an arbitrary number of children. In contrast to standard decision trees, items can take multiple paths and are assigned classes based on the weights that the paths encounter. Alternating decision trees can produce smaller and more interpretable classifiers than those obtained from applying boosting directly to standard decision trees.

Applications to computational biology

Decision trees have found wide application within computational biology and bioinformatics because of their usefulness for aggregating diverse types of data to make accurate predictions. Here we mention only a few of the many instances of their use.

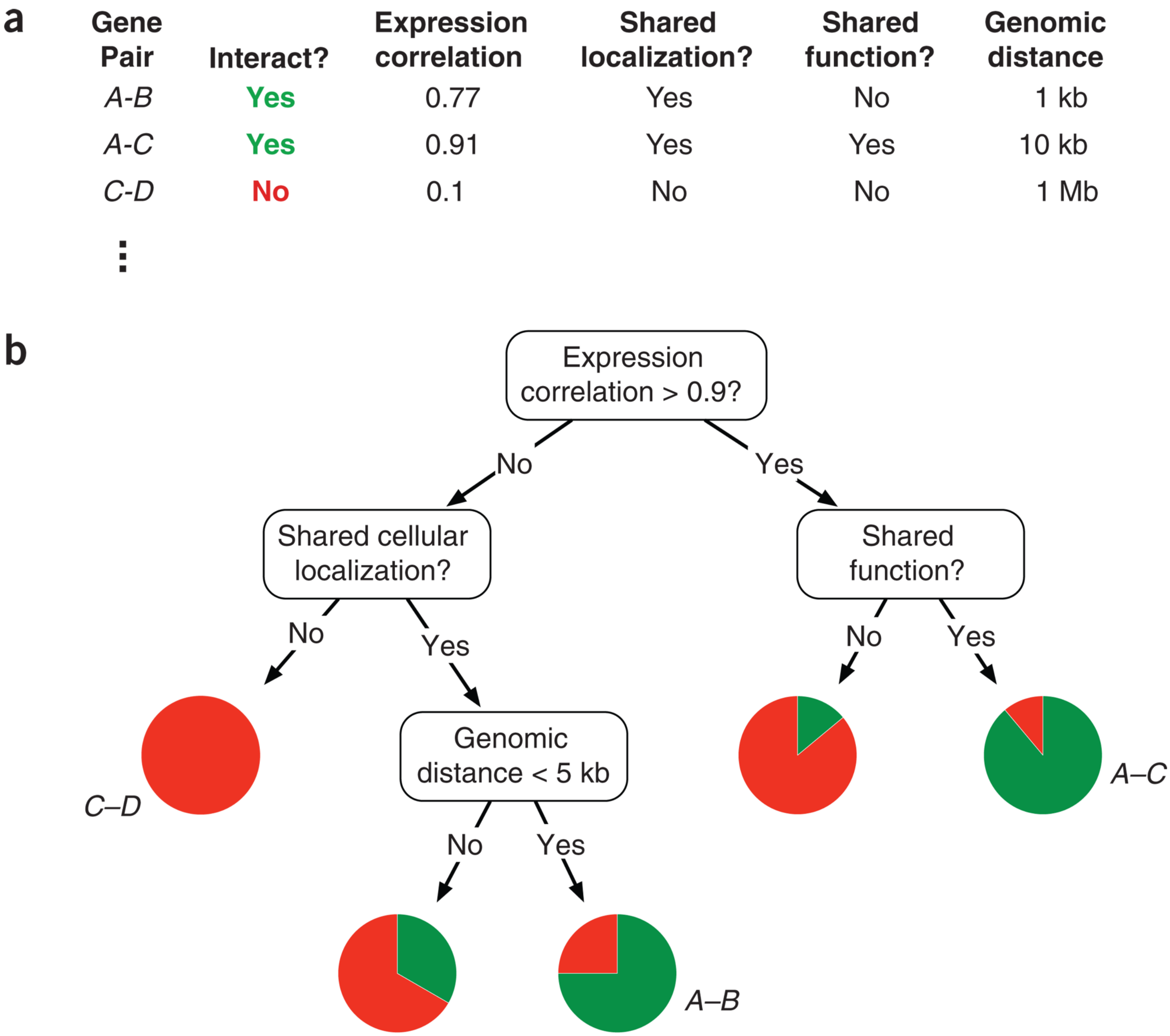
Synthetic sick and lethal (SSL) genetic interactions between genes A and B occur when the organism exhibits poor growth (or death) when both A and B are knocked out but not when either A or B is disabled individually. Wong *et al*.12 applied decision trees to predict SSL interactions in *Saccharomyces cerevisiae* using features as diverse as whether the two proteins interact physically, localize to the same place in the cell or have the function recorded in a database. They were able to identify a high percentage of SSL interactions with a low false-positive rate. In addition, analysis of the computed trees hinted at several mechanisms underlying SSL interactions.

Computational gene finders use a variety of approaches to determine the correct exonintron structure of eukaryotic genes. *Ab initio* gene finders use information inherent in the sequence, whereas alignment-based methods use sequence similarity among related species. Allen *et al*.13 used decision trees within the JIGSAW system to combine evidence from many different gene finding methods, resulting in an integrated method that is one of the best available ways to find genes in the human genome and the genomes of other species.

Middendorf *et al*.14 used alternating decision trees to predict whether an *S. cerevisiae* gene would be up- or downregulated under particular conditions of transcription regulator expression given the sequence of its regulatory region. In addition to good performance predicting the expression state of target genes, they were able to identify motifs and regulators that appear to control the expression of the target genes.

References

1. Quinlan, JR. C4.5: Programs for Machine Learning. San Mateo, CA, USA: Morgan Kaufmann Publishers; 1993.
2. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. Classification and Regression Trees. Belmont, CA, USA: Wadsworth International Group; 1984.
3. Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In: Cohen, WW.; Moore, A., editors. Machine Learning, Proceedings of the Twenty-Third International Conference; New York: ACM; 2003. p. 161-168.
4. Zadrozny, B.; Elkan, C. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: Brodley, CE.; Danyluk, AP., editors. Proceedings of the 18th International Conference on Machine Learning; San Francisco: Morgan Kaufmann; 2001. p. 609-616.
5. Murthy SK, Kasif S, Salzberg S. A system for induction of oblique decision trees. J. Artif. Intell. Res 1994;2:1–32.
6. MacKay, DJC. Information Theory, Inference and Learning Algorithms. Cambridge, UK: Cambridge University Press; 2003.
7. Quinlan JR, Rivest RL. Inferring decision trees using the minimum Description Length Principle. Inf. Comput 1989;80:227–248.
8. Breiman L. Random forests. Mach. Learn 2001;45:5–32.
9. Heath, D.; Kasif, S.; Salzberg, S. Committees of decision trees. In: Gorayska, B.; Mey, J., editors. Cognitive Technology: In Search of a Human Interface. Amsterdam, The Netherlands: Elsevier Science; 1996. p. 305-317.
10. Schapire, RE. The boosting approach to machine learning: an overview. In: Denison, DD.; Hansen, MH.; Holmes, CC.; Mallick, B.; Yu, B., editors. Nonlinear Estimation and Classification. New York: Springer; 2003. p. 141-171.
11. Freund, Y.; Mason, L. The alternating decision tree learning algorithm. In: Bratko, I.; Džeroski, S., editors. Proceedings of the 16th International Conference on Machine Learning; San Francisco: Morgan Kaufmann; 1999. p. 124-133.
12. Wong SL, et al. Combining biological networks to predict genetic interactions. Proc. Natl. Acad. Sci. USA 2004;101:15682–15687. [PubMed: 15496468]
13. Allen JE, Majoros WH, Pertea M, Salzberg SL. JIGSAW, GeneZilla, and GlimmerHMM: puzzling out the features of human genes in the ENCODE regions. Genome Biol 2006;7:S9. [PubMed: 16925843]
14. Middendorf M, Kundaje A, Wiggins C, Freund Y, Leslie C. Predicting genetic regulatory response using classification. Bioinformatics 2004;20:i232–i240. [PubMed: 15262804]
15. Chen X-W, Liu W. Prediction of protein-protein interactions using random decision forest framework. Bioinformatics 2005;21:4394–4400. [PubMed: 16234318]



a

Gene Pair	Interact?	Expression correlation	Shared localization?	Shared function?	Genomic distance
A-B	Yes	0.77	Yes	No	1 kb
A-C	Yes	0.91	Yes	Yes	10 kb
C-D	No	0.1	No	No	1 Mb
⋮					

Figure 1. A hypothetical example of how a decision tree might predict protein-protein interactions
(a) Each data item is a gene pair associated with a variety of features. Some features are real-valued numbers (such as the chromosomal distance between the genes or the correlation coefficient of their expression profiles under a set of conditions). Other features are categorical (such as whether the proteins co-localize or are annotated with the same function). Only a few training examples are shown. **(b)** A hypothetical decision tree in which each node contains a yes/no question asking about a single feature of the data items. An example arrives at a leaf according to the answers to the questions. Pie charts indicate the percentage of interactors (green) and noninteractors (red) from the training examples that reach each leaf. New examples are predicted to interact if they reach a predominately green leaf or to not interact if they reach a predominately red leaf. In practice, random forests have been used to predict protein-protein interactions15.