In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, 1997, pp. 846-851.

Decision Tree Grafting

Geoffrey I. Webb
School of Computing and Mathematics
Deakin University
Geelong, Vic, 3217, Australia.

Abstract

This paper extends recent work on decision tree grafting. Grafting is an inductive process that adds nodes to inferred decision trees. This process is demonstrated to frequently improve predictive accuracy. Superficial analysis might suggest that decision tree grafting is the direct reverse of pruning. To the contrary, it is argued that the two processes are complementary. This is because, like standard tree growing techniques, pruning uses only local information, whereas grafting uses non-local information. The use of both pruning and grafting in conjunction is demonstrated to provide the best general predictive accuracy over a representative selection of learning tasks.

1 Introduction

Decision tree pruning [Breiman *et al.*, 1984; Quinlan, 1987] is a widely accepted method for post-processing decision trees. Pruning removes nodes from an inferred decision tree. It has been demonstrated to improve the predictive accuracy of inferred decision trees in a wide variety of domains [Breiman *et al.*, 1984; Quinlan, 1987]. A classifier can be viewed as partitioning an instance space. Each partition associates a set of possible objects with a class. Pruning reduces the number of partitions imposed on an instance space by a decision tree.

In contrast to pruning, a number of recent studies have suggested that predictive accuracy may also be improved by more complex partitioning of an instance space than that formed by standard decision tree induction. Predictive accuracy has been improved both by:

- grafting additional leaves [Webb, 1996]; and
- developing multiple classifiers that are used in conjunction to classify objects [Ali *et al.*, 1994; Breiman, 1996; Dietterich and Bakiri, 1994; Kwok and Carter, 1990; Oliver and Hand, 1995; Nock and Gascuel, 1995; Schapire, 1990; Wolpert, 1992].

The latter approaches lead to complex implicit partitioning of the instance space through resolution of the conflicts between the individual classifiers' partitions. Direct grafting forms an explicit representation of the final partitioning of the instance space by adding new branches to a decision tree after the completion of conventional decision tree induction.

The increase in predictive accuracy resulting from more complex partitioning of the instance space can be explained as follows. Conventional machine learning techniques consider only areas of the instance space directly occupied by training examples. Areas of the instance space that are not occupied by training examples are assigned to partitions as a side-effect of partitioning occupied areas. This occurs without consideration of the available evidence relating to appropriate partitioning of these regions. Explicit examination of such areas may provide evidence as to the most likely class for previously unseen objects that fall therein. If there is such evidence and the appropriate classification differs from that currently assigned to the region, a new partition can be formed. This is achieved by grafting a new leaf onto the tree.

The use of multiple classifiers obtains this result in a more indirect manner. Each classifier will form different partitions. Regions occupied by no training examples may fall within different partitions for each classifier. The strength of evidence associated with that region for each classifier can be evaluated and a most highly supported prediction made.

Consider an abstract example (Figure 1). This illustrates a simple instance space occupied by objects of three classes ($*, \bullet$ and $\diamond$). Objects are described by two attributes A and B. These attributes define a two dimensional instance space. An instance of unknown class is also depicted (?). On visual inspection it is plausible that this unknown case belongs to class $\diamond$ as it is close to a number of instances of this class. However, most decision tree learners would create a partition that assigned this point to class $*$. Figure 2 indicates the partitions

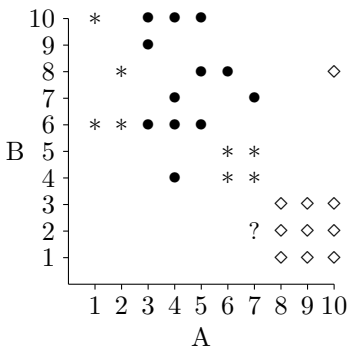

Figure 1: Example instance space

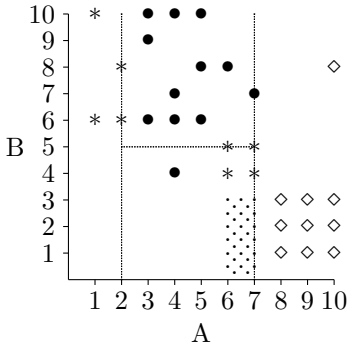

Figure 2: Example instance space as partitioned by C4.5

created by C4.5 [Quinlan, 1993], a pre-eminent example of a decision tree learner. In contrast, it is plausible to assign the shaded region to class $\diamond$. The C4.5x [Webb, 1996] grafting procedure identifies such regions and grafts new leaves onto the decision tree to form appropriate new partitions of the instance space.

The primary focus of Webb's [1996] grafting research was to examine the effect of complexity on predictive accuracy. Consequently, C4.5x was designed to control other potential confounding factors, specifically resubstitution performance. These measures could reduce the predictive accuracy of the inferred trees [Webb, 1996].

This paper seeks to extend Webb's [1996] grafting research by developing grafting techniques aimed to maximize predictive accuracy. Four key changes to the C4.5x approach are presented: allowing grafting to alter resubstitution performance; the ordered addition of multiple new branches in the place of a single original leaf; the use of a significance test to restrict the selection of new branches; and allowing grafting within leaves occupied by no training examples.

Evaluation on twenty representative learning domains demonstrates that the application of the new techniques frequently results in the induction of decision trees with improved predictive accuracy.

2 Techniques for decision tree grafting

The new post-processor, C4.5+, operates by examining each leaf l of an inferred tree in turn. It climbs the tree examining each ancestor node n for evidence supporting alternative partitions within l. This evidence is obtained by considering cuts that could have been employed at n, that would provide stronger evidence in support of a particular class dominating a region within l than that provided by the distribution of objects at l. In doing so, it only considers cuts that fall within the range of values for an attribute that can reach l. It also excludes from consideration cuts that would reclassify an object at l that is correctly classified by l. A set of such cuts are assembled. These are used to graft new branches and leaves onto the decision tree between l and its parent. At present there is no consideration of potential new branches on discrete valued attributes, although in principle this should be straight forward.

The evidence in support of each cut is evaluated using a Laplacian accuracy estimate [Niblett and Bratko, 1986]. Because each leaf relates to a binary classification (an object belongs to the class in question or does not), the binary form of Laplace is used. For threshold t on attribute a at leaf l, the evidence in support of labeling the partition below t with class x is the maximum value for an ancestor node n of l for the formula $\frac{P+1}{T+2}$ where T is the number of objects at n for which $min < a \leq t$; and P is the number of those objects that belong to class x. Calculation of the evidence in support of labeling a partition above a threshold differs only in that the objects for which $t < a \leq max$ are instead considered. Where l contains no training objects, it is treated as containing all objects at its parent for the sake of these calculations.

The best such $\leq$ and $>$ cut for each attribute is determined. A list of all these cuts is created, C.

The strength of evidence in support of the current labeling of l is calculated using the Laplace accuracy estimate considering the objects at l, where T is the number of objects at l and P is the number of those objects that belong to the class with which l is labeled.

Any cuts that do not have greater support than that for l are removed from C. A binomial test is also employed to further remove from C cuts for which there is insufficient evidence that the resulting leaf is drawn from a better distribution of examples than the original leaf (see Step 3 of the algorithm presented in Appendix A). C is sorted from the cut with highest support to that with lowest support. Trailing elements of C that support the creation of new leaves for the same class as l are deleted as they will not alter the tree's classifications. Then the cuts in C are inserted in order creating a sequence of new branches and leaves between l's parent and l.

This approach ensures that all new partitions define true regions. That is, for any attribute a and value v it is

not possible to partition on $a \leq v$ unless it is possible for both objects from the *domain* with values of a greater than v and objects with values less than or equal to v to reach the node being partitioned (even though it is possible that no objects from the *training set* will fall within the new partition). In particular, this ensures that new cuts are not simple duplications of existing cuts at ancestors to the current node. Thus, every modification adds non-redundant complexity to the tree.

This algorithm is presented in Appendix A. C4.5+ differs from C4.5x [Webb, 1996] by

1. adding multiple leaves at each original leaf — C4.5x added the new leaf with maximal support only;
2. using a binomial test to prevent the addition of leaves for which there is insufficient evidence that the leaf is drawn from a better distribution of examples (Algorithm Step 3);
3. allowing new leaves to reclassify training examples (although only if those examples are misclassified by the original leaf); and
4. using the training examples at the parent node when a leaf has no training examples — C4.5x did not allow grafting additional leaves onto an existing leaf that covers no training examples.

Adding multiple leaves can be expected to be beneficial as every piece of additional evidence can be utilized. However, initial experimentation suggested that adding leaves for which the level of additional support was marginal, while often beneficial, could also often reduce predictive accuracy. The use of a binomial test to evaluate the comparative strength of support for a new leaf is intended to reduce the risk of adding leaves that appear better by chance alone.

Allowing new leaves to reclassify training examples has intuitive appeal. If there is evidence that a region of the instance space should be associated with a given class, the existence of an object of that class in that region should not prevent a system from forming that association. For example, the object at $A = 4, B = 4$ in Figure 1 should not stop C4.5+ from relabeling that region as belonging to $\bullet$. C4.5x prohibited such grafting actions to avoid experimental confounds arising from differing resubstitution accuracy between treatments [Webb, 1996].

The training examples from the parent node are used for leaves that cover no training examples, as the parent node provides the best available evidence of the class distribution in the neighborhood of the leaf. Such leaves are prime candidates for modification as the local evidence in support of any given class assignment is unlikely to be strong.

3 Example

C4.5 creates the following decision tree for the example training set illustrated in Figure 1. The partitions created by this tree are illustrated in Figure 2.

```
A > 7 :  ◇ (10.0)
A <= 7 :
| A <= 2 :  ∗ (4.0)
| A > 2 :
| | B <= 5 :  ∗ (5.0/1.0)
| | B > 5 :  • (11.0)
```

C4.5 has placed cuts on A at values 7 and 2. $A > 7$ is assigned to class $\diamond$ and $A \leq 2$ is assigned to $*$. In the region for $2 < A \leq 7$, a cut has been placed at $B = 5$. $B \leq 5$ has been assigned to $*$ and $B > 5$ to $\bullet$. The figures in brackets indicate the numbers of training examples at each leaf and, where applicable, the number of these that are misclassified.

C4.5+ examines each of the 4 leaves in turn. At each leaf it climbs the tree looking for cuts that would not reclassify any training examples correctly classified at the leaf. The leaf for $A > 7$ has only the root as an ancestor. No better cuts can be found.

To process the leaf for $A \leq 2$ the system climbs to its parent node, at which no better cuts can be found, and then to the root. At the root, all values are considered on both attributes that are greater or less than those of the (in all cases correctly classified) training examples from the leaf. There are no training examples with lower values on A or with greater values on B than those of the examples at the leaf. Values on A greater than those at the leaf are not considered as such a cut imposed at the leaf would define a new region of zero volume. All values are considered on B less than 6, the lowest value for an example at the leaf. A cut at 5 results in a partition containing 1 $\bullet$, 4 $*$ and 9 $\diamond$. The Laplace accuracy estimate for the region $B \leq 5$ for the majority class $\diamond$ is $\frac{9+1}{14+2} = 0.625$. The class distributions and accuracy estimates of the remaining possible cuts are:

4: 1/2/9, $\frac{9+1}{12+2} = 0.714$;

3: 0/0/9, $\frac{9+1}{9+2} = 0.909$;

2: 0/0/6, $\frac{6+1}{6+2} = 0.875$; and

1: 0/0/3, $\frac{3+1}{3+2} = 0.800$.

The best of these cuts is at value 3 with accuracy estimate 0.909. The original leaf is occupied by four points all of which are correctly classified resulting in an accuracy estimate of $\frac{4+1}{4+2} = 0.833$. The probability of obtaining the class distribution (9 positive and 0 negative) given the estimated accuracy for the original leaf (0.833) is less than 0.05, so the selected cut is grafted between the original leaf and its parent. The dominating class ($\diamond$) for the new region in the ancestor node from which the evidence was obtained is assigned to the new leaf.

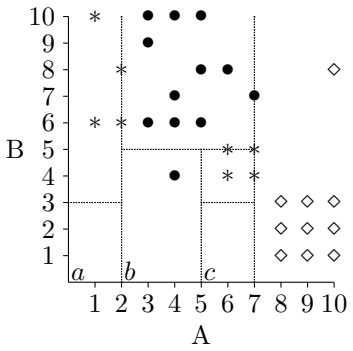

Figure 3: Example instance space after grafting

Next the system considers the leaf below the branch $B \leq 5$. The accuracy estimate at this leaf is $\frac{4+1}{5+2} = 0.714$. At the parent node (the node reached by the branches $A \leq 7$ then $A > 2$), a cut at $A = 5$ creates a leaf containing 10 $\bullet$ and no examples of other classes. The resulting accuracy estimate is 0.917. The probability of obtaining this distribution given the estimated accuracy for the leaf is less than 0.05, so the new cut is accepted. Another cut, at $B = 3$, is found at the root. The partition formed by this cut contains 9 $\diamond$ and no other examples. The resulting accuracy estimate is $\frac{9+1}{9+2} = 0.909$ The probability of obtaining this class distribution given the estimated accuracy at the original leaf is also less than 0.05. In consequence, this cut is also accepted. Other potential $\leq$ cuts on these attributes receive lower accuracy estimates and so are discarded. Branches for the two cuts are grafted in order of their accuracy estimate.

No appropriate new cuts can be found for the leaf below $B > 5$.

The partitions imposed by the resulting tree are illustrated in Figure 3. The new partitions labeled a and c are assigned to $\diamond$ and partition b to $*$. While partition a may have less intuitive support than b or c, the support for any classification within this region is weak and the class $\diamond$ is at least as plausible as either alternative.

4 Experimental evaluation

The postprocessing algorithm was implemented as an extension to C4.5 Release 8 [Quinlan, 1993].

It was evaluated by application to twenty representative learning tasks from the UCI Machine Learning Repository. These datasets are described in Table 1. They show considerable diversity in size, number of classes, and type and number of attributes, within the restriction that all contain continuous attributes, as these are the only attributes on which grafting is implemented.

Three variants of the system were tested. **All** included the full system as described in Appendix A. **None** was C4.5 release 8 with no post-processing. **One** added at most one new leaf to each existing leaf. This was achieved by discarding all but the highest valued tuple after Step 3.

Table 1: Description of data sets

Name	Cases	Classes	Contin.	Discr.
anneal	898	6	6	32
balance-scale	625	3	4	—
breast-wisc	699	2	9	—
cleveland-hd	303	2	6	7
crx	690	2	6	9
dis	3772	2	7	22
echocardiogram	74	2	5	1
german-credit	1000	2	7	13
glass	214	7	9	—
hepatitis	155	2	6	13
horse-colic	368	2	8	13
hungarian-hd	294	2	6	7
hypo	3772	4	7	22
iris	150	3	4	—
labor-neg	57	2	8	8
new-thyroid	215	3	5	—
Pima-diabetes	768	2	8	—
sick	3772	2	7	22
sonar	208	2	60	—
waveform	300	3	21	—

C4.5 employs a two stage process to infer decision trees from data. An initial unpruned tree is created. This is then simplified to produce a pruned tree. Each variant of the post-processing algorithm was used to post-process both pruned and unpruned trees produced by C4.5.

Ten stratified ten-fold cross validation experiments were performed for each data set. In each of these experiments, the data set was divided into ten subsets of as close as possible to equal size with as close as possible to identical class distributions. For each subset, each treatment was applied to learn a decision tree from all the remaining subsets, and then applied to predict the class of each object in the selected subset.

Table 2 presents the predictive accuracy obtained for each treatment in these experiments. The mean percentage error over all one hundred sets of predictions is presented for each treatment. Two summary lines present for each of the other treatments

- a win-loss summary of the number of data sets for which the mean error is lower or higher than that of *all*; and
- the one-tailed binomial probability of obtaining such a win-loss result by chance.

It can be seen that *all* has lower error than *none* significantly (at the 0.05 level) more often both for pruned and unpruned trees. However, the advantage to *all* over *one* is not significant at the 0.05 level.

The magnitude of the changes also differs greatly. The largest increase in error resulting from the addition of all grafts is 1.0% for the iris data. The largest reduction in

Table 2: Summary of mean percentage error rates

	Pruned Trees			Unpruned Trees		
	All	**None**	**One**	**All**	**None**	**One**
anneal	7.2	7.5	7.4	5.3	5.5	5.4

Table 3: Summary of mean resubstitution error rates

	Pruned Trees			Unpruned Trees		
	All	**None**	**One**	**All**	**None**	**One**
anneal	5.1	5.4	5.3	2.9	3.1	3.1

Table 4: Summary of mean number of nodes per tree

	Pruned Trees			Unpruned Trees		
	All	**None**	**One**	**All**	**None**	**One**
anneal	350.7	78.6	120.8	562.8	141.4	209.7

error is 2.9% for unpruned trees on the hepatitis data. The postprocessing of pruned trees results in reductions of 1.0% or more for seven of the twenty datasets.

It is interesting to compare the performance of post-processing both pruned and unpruned trees. Pruning then grafting produces lower error than grafting alone for twelve data sets whereas the reverse is true for only three. A one-tailed binomial sign test reveals that this difference is significant at the 0.05 level ($p = 0.017$). It appears that both pruning and grafting have a valuable role to play in decision tree induction. It is possible that this results from the abilities of pruning to identify partitions where the local information is insufficient to create sensible sub-partitions and of grafting to use non-local information to then create suitable sub-partitions. The reduction in resubstitution error brought about by grafting (Table 3) lends some support to this explanation.

Table 4 presents the number of nodes obtained by each treatment employing the same format as in Table 2. Adding all nodes produces more complex trees than either of the other treatments for every data set.

5 Conclusions

The experimental results suggest that C4.5+ is successful in identifying regions of the instance space occupied by no training examples for which initial tree induction has made poor class choices. Grafting new nodes to correct these poor class assignments can significantly improve the predictive accuracy of the inferred decision trees. The extension of the techniques to graft multiple new branches at each leaf of the original tree led to more reductions than increases in error when compared to the C4.5x technique of adding at most one new branch per leaf. However, the frequency with which the addition of more branches increases error and the failure to obtain a statistically significant advantage in this respect suggests that there is room for further improvement in the filtering that is used to select which of the potential new branches should be grafted to the tree.

Research on grafting to date has examined only the addition of tests on continuous attributes. The techniques should extend in a straight forward manner to discrete attributes. The development of appropriate grafting techniques for discrete attributes is a promising di-

rection for future research.

The application of both grafting and pruning results in lower average error significantly more often than does grafting alone. It is possible that this is due to the ability of pruning to identify partitions of the instance space where the local information is insufficient to create sensible sub-partitions. Grafting can then use non-local information to generate appropriate sub-partitions.

However, many benefits have counterweighing costs and grafting is no exception. The increase in accuracy obtained through grafting is often modest. This is obtained at the expense of large increases in decision tree complexity. In applications where classifier complexity is a significant factor, this trade-off deserves careful consideration before grafting is employed.

It has been argued herein that grafting has a similar effect to the induction and application of multiple classifiers, with the difference that grafting incorporates its complex instance space partitioning into a single explicit decision tree instead of requiring the resolution of multiple distinct partitionings to determine the ultimate underlying partitioning to be applied. Exploration of this hypothesized relationship provides further promising avenues for future research.

A Algorithm

Let $cases(n)$ denote the set of all training examples that can reach node n, unless there are no such examples in which case $cases(n)$ shall denote the set of all training examples that can reach the parent of n.

Let $value(a, x)$ denote the value of attribute a for training example x.

Let $pos(X, c)$ denote the number of objects of class c in the set of training examples X.

Let $class(x)$ denote the class of object x.

Let $Laplace(X, c) = \frac{pos(X,c)+1}{|X|+2}$ where X is a set of training examples, $|X|$ is the number of training examples and c is a class.

Let $upperlim(n, a)$ denote the minimum value of a cut c on attribute a for an ancestor node x of n with respect to which n lies below the $a \leq c$ branch of x. If there is no such cut, $upperlim(n, a) = \infty$. This determines an upper bound on the values for a that may reach n.

Let $lowerlim(n, a)$ denote the maximum value of a cut c on attribute a for an ancestor node x of n with respect to which n lies below the $a > c$ branch of x. If there is no such cut, $lowerlim(n, a) = -\infty$. This determines a lower bound on the values for a that may reach n.

Let $prob(x, n, p)$ be the probability of obtaining x or more positive objects in a random selection of n objects if the probability of selecting a positive object is p.

To post-process leaf l dominated by class c

1. Initialize to {} a set of tuples t representing potential cuts.
2. For each continuous attribute a
   (a) Find values of
   n: n is an ancestor of l
   v: $\exists x$: $x \in cases(n)$ & $v = value(a, x)$ & $v \leq min(value(a, y)$: $y \in cases(l)$ & $class(y) = c)$ & $v > lowerlim(l, a)$
   k: k is a class
   that maximize $\mathcal{L}' = Laplace(\{x$: $x \in cases(n)$ & $value(a, x) \leq v$ & $value(a, x) > lowerlim(l, a)\}, k)$.
   (b) Add to t the tuple $\langle n, a, v, k, \mathcal{L}', \leq \rangle$
   (c) Find values of
   n: n is an ancestor of l
   v: $\exists x$: $x \in cases(n)$ & $v = value(a, x)$ & $v > max(value(a, y)$: $y \in cases(l)$ & $class(y) = c)$ & $v \leq upperlim(l, a)$
   k: k is a class
   that maximize $\mathcal{L}' = Laplace(\{x$: $x \in cases(n)$ & $value(a, x) > v$ & $value(a, x) \leq upperlim(l, a)\}, k)$.
   (d) Add to t the tuple $\langle n, a, v, k, \mathcal{L}', > \rangle$
3. Remove from t all tuples $\langle n, a, v, k, \mathcal{L}, x \rangle$ such that $\mathcal{L} \leq Laplace(cases(l), c)$ or $prob(x, n, Laplace(cases(l), c)) \leq 0.05$.
4. Remove from t all tuples $\langle n, a, v, c, \mathcal{L}, x \rangle$ such that there is no tuple $\langle n', a', v', k', \mathcal{L}', x' \rangle$ such that $k' \neq c$ & $\mathcal{L}' < \mathcal{L}$.
5. For each $\langle n, a, v, k, \mathcal{L}, x \rangle$ in t ordered on L from highest to lowest value
   If x is $\leq$ then
   (a) replace l with a node n with the test $a \leq v$.
   (b) set the $\leq$ branch for n to lead to a leaf for class k.
   (c) set the $>$ branch for n to lead to l.
   else (x must be $>$)
   (a) replace l with a node n with the test $a \leq v$.
   (b) set the $>$ branch for n to lead to a leaf for class k.
   (c) set the $\leq$ branch for n to lead to l.

References

[Ali *et al.*, 1994] Kamal Ali, Clifford Brunk, and Michael Pazzani. On learning multiple descriptions of a concept. In *Proceedings of Tools with Artificial Intelligence*, pages 476–483, New Orleans, LA, 1994.

[Breiman *et al.*, 1984] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth International, Belmont, CA, 1984.

[Breiman, 1996] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[Dietterich and Bakiri, 1994] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286., 1994.

[Kwok and Carter, 1990] Suk Wah Kwok and Chris Carter. Multiple decision trees. In R. D. Shachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 4*, pages 327–335. North Holland, Amsterdam, 1990.

[Niblett and Bratko, 1986] Tim Niblett and Ivan Bratko. Learning decision rules in noisy domains. In M. A. Bramer, editor, *Research and Development in Expert Systems III*, pages 25–34. Cambridge University Press, Cambridge, 1986.

[Nock and Gascuel, 1995] Richard Nock and Olivier Gascuel. On learning decision committees. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 413–420, Taho City, CA, July 1995. Morgan Kaufmann.

[Oliver and Hand, 1995] Jonathon J. Oliver and David J. Hand. On pruning and averaging decision trees. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 430–437, Taho City, CA, July 1995. Morgan Kaufmann.

[Quinlan, 1987] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Schapire, 1990] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[Webb, 1996] Geoffrey I. Webb. Further experimental evidence against the utility of Occam's razor. *Journal of Artificial Intelligence Research*, 4:397–417, 1996.

[Wolpert, 1992] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.