Proc. IJCAI, 97.

Unbiased Assessment of Learning Algorithms

Tobias Scheffer and **Ralf Herbrich**
Technische Universität Berlin, Artificial Intelligence Group, FR 5-8
Franklinstr. 28/29, 10587 Berlin, Germany
scheffer@cs.tu-berlin.de

Abstract

In order to rank the performance of machine learning algorithms, many researchers conduct experiments on benchmark data sets. Since most learning algorithms have domain-specific parameters, it is a popular custom to adapt these parameters to obtain a minimal error rate on the test set. The same rate is then used to rank the algorithm, which causes an optimistic bias. We quantify this bias, showing, in particular, that an algorithm with more parameters will probably be ranked higher than an equally good algorithm with fewer parameters. We demonstrate this result, showing the number of parameters and trials required in order to pretend to outperform C4.5 or FOIL, respectively, for various benchmark problems. We then describe out how unbiased ranking experiments should be conducted.

1 Introduction

Estimating the accuracy of a classifier is a topic that has experienced much attention in the ML community. One of the main results is that N-fold cross validation provides a bias-free [Sto74] though not variance-free [Zha92; Koh95], estimate of the true accuracy. n-fold cross validation means that n classifiers are learned from $((n-1)/n)$ths of the available data, and tested on the remaining $(1/n)$th of the training set. The averaged accuracies are a bias-free estimate of the accuracy of a classifier that is learned by the same algorithm on the complete data set. If the data set is too large, the accuracy is usually estimated on a test set that was not used for learning (one-shot training and test), which causes a slight pessimistic bias. For model selection purposes, a .632 bootstrap [Efr79] may be preferable. Bootstrap experiments are conducted by re-sampling a number of training sets of size n from an original data set of size n by randomly drawing samples *with replacement.* On the average, $(1-1/e)m \approx .632m$ distinct samples will appear in the training set, and the averaged accuracies on the remaining test sets provide an optimistically biased estimate. The variance is claimed to be lower in many cases than the variance of cross validation [Efr83], which is important when choosing an optimal model.

Many papers propose new or modified machine learning algorithms, with claims such as "my new algorithm B is way better than algorithm A", or "extension X improves algorithm A a lot" typically supported by ranking experiments on a well known set of benchmark problems. There is also a book [MST94], resulting from the European StatLog project, that is dedicated to the comparison of learning algorithms for benchmark problems.

Virtually any learning algorithm possesses a number of parameters (*e.g.*, learning rates, number of learning steps, pruning thresholds, etc.). Selecting values for these parameters is the model selection task, which has to be considered a part of the training process. Unfortunately, it has become custom to adjust these parameters such that the error on either the test set, or, in case of n-fold cross validation, the averaged error on the n test sets, is minimized. Since the error on the test set is used as the quality criterion for the model selection task, the test set influences the training process. Hence, the assumption that the test sets are not used for learning, which is essential to the result that n-fold cross validation is bias-free, is violated.

Many authors are aware of that problem and properly separate model selection from accuracy estimation, *e.g.,* [KJ97], but a majority of authors seem to consider the resulting distortion of the results negligible. One of many examples is the StatLog project [MST94], where it has not been taken into account by all contributing partners. A slight bias would not be dramatic, if all learning algorithms would be effected equally, such that ranking results would still remain valid. But we argue that the parameters form a communication channel from the test set to the learning algorithm (see Figure 1), and that the resulting bias depends on the capacity of this chan-

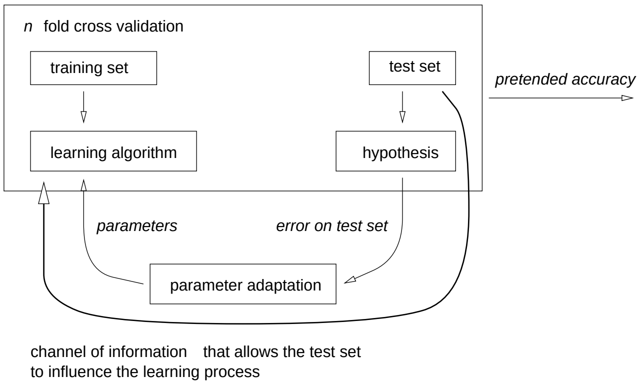

Figure 1: If model selection and accuracy estimation are mixed, the parameters for a communication channel, delivering information about the test set to the learning algorithm

nel, *i.e.,* the number of parameters, and the number of different parameter settings which are tested. The main reason why many authors conduct their experiments in the naive way is that unbiased experiments imply a lot of additional computational effort, especially when the parameters are manually adjusted. Our main contribution is to quantify this parameter-bias, identifying some cases where it actually is negligible and showing others where it makes ranking results invalid.

In the following sections, we consider different experimental settings: Section 2 quantifies the bias of one-shot training and test, Section 3 quantifies the bias of n-fold cross validation when different parameters are used for each of the n trials, and Section 4 is dedicated to cross validation with equal parameter settings for each of the n trials. Section 5 finally recalls the proper way of conducting ranking experiments.

2 One-shot training and test

In this section, we assume the following setting. A learning algorithm L accepts a set of parameters and there are 2^p distinct parameter settings (*i.e.,* the algorithm possesses $log_b(2^p)$ parameters with b possible values each). This set of parameters can be viewed as a communication channel from the parameter optimizer to the learner with a capacity of p bits.

2.1 Communication

As Figure 1 illustrates, when the learner is presented a set of parameters and a training set, it generates a hypothesis h which is used to determine the accuracy of h on the test set of size m. The parameter optimizer is told this accuracy and responds with a new set of parameters, which are again used for training. This cycle is repeated t times, and the best observed accuracy on the test set is then submitted for publication. Based on the accuracy measured on the test set, the parameter optimizer can send p bits of information on the test set to the learning algorithm.

If H is the entropy of the test set then the capacity of the parameter channel allows to transmit information about the class labels of $c = \frac{p}{H}$ test samples (because H bits are required to encode the class label of one sample), *e.g.*, if the test set contains 4 uniformly-distributed class labels, the entropy will be 2, allowing a parameter channel of width 4 (16 distinct settings) to transmit the class labels of $4/2 = 2$ test objects. This leaves two questions to be answered: how does the parameter optimizer get to know the class labels of the first c test instances, and how much will this knowledge improve the result?

2.2 Parameter adjustment

The parameter optimizer guesses parameters and obtains the accuracy on the test set in return. We assume that the learning algorithm passes these parameters to the hypothesis, which uses the information to classify the first c test objects that it encounters in some particular way, *i.e.,*the parameters encode class labels for the first c test objects. Then the parameter optimizer can use the following strategy:

1. for all objects i, 1 through c
   (a) for all labels l, 1 through $|classes|$
      – tell the learner to classify the ith object as class l and determine the accuracy acc_l
   (b) keep the label of i to the class value $l*$ that resulted in the highest accuracy acc_{l*}

This algorithm tries the assignment of every possible class label to each of c test samples, resulting in a complexity of $c \cdot |classes|$ learning trials, and finds the parameter setting that encodes correct class labels for c samples. We can view this algorithm as greedy search for optimal learning parameters, but since the correct class label of sample i is independent of the assigned class label of any other sample, the greedy algorithm will find an optimal assignment after $c \cdot |classes|$ trials.[1]

2.3 Increase in observed accuracy

Now, after being told the class labels of the first c test objects, how much can we hope to improve the accuracy on the test set? Assume that we have an initial

[1]Note that this is not a lower bound for the number of trials, as we have found an algorithm that needs only $c(\frac{3}{2} - \frac{1}{2}c + n) - n$ trials in the worst case, but while the algorithm given above essentially performs a gradient search in parameter space, the faster algorithm behaves unlike we would expect a parameter optimizer to behave, so the first result should be somewhat closer to the behavior of a "real" learning algorithm.

classifier h, learned by C4.5 [Qui92] say, which classifies p_h objects of the test set (of size m) correctly. For the first c test instances we repeat the class labels that we were told by the parameter optimizer, rather than using h, we classify the remaining $m-c$ instances using h. Hence the number of hits is $p_h - x + c$, where x is the number of hits that h would have obtained on the first c test instances (which are now classified correctly). We now determine the probability $P(c-X \geq z)$ that this procedure increases the number of hits by z. Drawing c samples from a total of m, we know that p_h of the m are "hits" (classified correctly by h). Hence, the number of hits within the c drawn samples follows a hyper-geometric distribution (note that m is finite). We then replace the c drawn samples with c "hits" (because the parameter optimizer "tells" us their class labels via the parameter channel).

$$\begin{aligned} P(c-X \geq z) &= \sum_{k=z}^{c} P(c-X=k) & (1)\\ &= \sum_{k=z}^{c} P(X=c-k) & (2)\\ &= \sum_{k=z}^{c} H(m,p_h,c)(c-k) & (3)\\ &= \sum_{k=z}^{c} \frac{\binom{p_h}{c-k}\binom{m-p_h}{k}}{\binom{m}{c}} & (4) \end{aligned}$$

This leads us to the bias and computational effort (explained in Section 2.2), required to achieve this bias for one-shot training and test. t is the expected number of learning experiments that need to be conducted in order to obtain $p_h + z$ hits on the m test samples with probability $P(c-X \geq z)$, when p_h is the true hit rate of h, provided a parameter channel with p bits of capacity is used.

$$P(c-X \geq z) = \sum_{k=z}^{\frac{p}{H}} \frac{\binom{p_h}{\frac{p}{H}-k}\binom{m-p_h}{k}}{\binom{m}{\frac{p}{H}}} \quad (5)$$

$$t = |classes| \cdot \frac{p}{H} \quad (6)$$

2.4 Affected benchmark problems

In this section, as well as in Sections 3.1 and 4.1, we will quantify the bias on concrete data sets. We assume that we use a real learning algorithm, C4.5 in most cases, which we "tune" with additional parameters in order to pretend to outperform the learning algorithm one rank higher than the initial learner. We will answer two questions: How many parameters do we need to succeed with probability $> 90\%$ after performing sufficiently many trials (note that this is an exact rather than an empirical result) and how many trials do we need in the average to obtain this result, provided the parameter optimizer performs gradient search in parameter space, which may be an inexact result since it depends on the actual optimizer as well as how strong the parameters influence the result.

Land-sat satellite images: This data set contains 4435 training and 2000 test instances. The default error rate is .231. Based on [MST94], C4.5 is ranked 10th (error .150, *i.e.,*1700 hits on the test set). To be ranked 9th it would have to outperform Bay-tree (error .147) for which C4.5 needs only $z=6$ extra hits on the test set. If we get class labels of $c=60$ instances, then $P(c-X \geq z) > 90\%$. Hence we need $t = 6 \cdot 60 = 360$ trials with different parameter settings and since $H=2.5$ we need a parameter channel of 150 bits. Although an automatic parameter adjustment system may very well run 360 trials, a parameter channel of 150 bits is fairly uncommon (*e.g.,* achieved by 45 parameters with 10 possible values each). Using $c=16$ samples (parameter channel of 40 bits) and 96 experiments, we still have a 2% chance of being over-ranked.

DNA: This data set, also described in [MST94], possesses 2000 training and 1186 test instances. C4.5 is ranked 10th (1096 hits). To be ranked 9th it would have to outperform INDCart, requiring $z=4$ extra hits. We need to be told $c=85$ class labels to achieve this with 90%: $P(c-X \geq z) > .9$. Since we have 3 classes, we need $t = 3 \cdot 85 = 255$ trials. Since $H = 1.4908$, we need 126 bit of parameters, *e.g.,* 37 parameters with 10 possible values each.

For both data sets we conclude that only a very eager scientist may obtain a modest over-ranking of his algorithm by using an incremental algorithm and an automatic parameter-adjustment procedure.

3 n-fold cross validation with parameter adjustment

In this section we study the bias caused by parameter adaptation when n-fold cross validation is conducted with parameter values chosen differently for the n runs of the learning algorithm. As an example, the number of learning steps is crucial to the performance of back propagation. Often, the optimal number of learning steps is determined by observing the error on the test set and selecting the point at which the error rate starts increasing again. The minimum errors that occured in the n learning curves are then averaged and published, *i.e.,*the number of learning steps may not be fixed to one value within the n folds. Also, this setting is often used when the splits of the training set are explicitly stated (*e.g.,* mesh or vehicle silhouettes).

To achieve an average of z extra hits per fold this way, $n \cdot z$ has to equal at least $nc - \sum_i X_i$, where X_i is the number of hits lost by not using h on c samples of fold

i.

$$P(nc - \sum_{i=1}^{n} X_i \geq nz) \quad (7)$$
$$= \sum_{k=nz}^{nc} P(nc - \sum_{i=1}^{n} X_i = k) \quad (8)$$
$$= \sum_{k=nz}^{nc} P(\sum_{i=1}^{n} X_i = nc - k) \quad (9)$$

Unfortunately, there is no explicit formula for the distribution of a sum of hyper-geometric random numbers. However, we can split the probability of a sum of random numbers considering every possible combination that yields this sum [Ren70]: $P(X+Y=z) = \sum_j P(X=j)P(Y=z-j)$. We can use this equation to split the last summand, resulting in the following recursive equation: $P(\sum_{i=1}^{m} X_i = x) = \sum_{j=0}^{x} P(\sum_{i=1}^{m-1} X_i = x-j)P(X_m = j)$. Instantiated to our situation this yields

$$P(\sum_{i=1}^{n} X_i = nc - k) \quad (10)$$
$$= \sum_{l=0}^{nc-k} P(\sum_{i=1}^{n-1} X_i = nc - k - l) \cdot P(X_n = l) \quad (11)$$
$$= \sum_{l=0}^{nc-k} P(\sum_{i=1}^{n-1} X_i = nc - k - l) \cdot H(m, p_h, c)(l) \quad (12)$$
$$= \sum_{l=0}^{nc-k} P(\sum_{i=1}^{n-1} X_i = nc - k - l) \cdot \frac{\binom{p_h}{l}\binom{m-p_h}{c-l}}{\binom{m}{c}} \quad (13)$$

Straightforward evaluation of this recursive equation for an instantiation is very expensive, since for each evaluation of $P(\sum_{i=1}^{n} X_i = k)$ we must calculate $P(\sum_{i=1}^{n-1} X_i = k')$ for every $k' < k$. By iteratively filling an array that is indexed n and k with $P(\sum_{i=1}^{n} X_i = k)$ the formula can be evaluated quickly.

3.1 Affected benchmark problems

FEM mesh design: [DM94], a relational problem popular in inductive logic programming. It is explicitly split into five learning problems. There are 277 samples and 13 classes and the entropy is $H = 2.87$. FOIL [Qui90] achieves an accuracy of 21% (59 hits). To achieve 26% accuracy with a probability of 99%, FOIL needs $c = 5$ class labels, while to achieve 31% with a probability of 93% FOIL would need $c = 8$ class labels. To achieve 26%, we need to conduct 65 trials and a parameter channel of 14 bits, while to achieve 31% we require 104 trials and a parameter channel of 23 bits.of The parameter adaptation bias is so strong, that accuracy results can easily be pushed from 21% to 31% on this data set.

Diabetes: There are 768 samples with 2 classes, $H = .9331$. C4.5 achieves 561 hits (rank 13), we need to outperform Quadisc (.5 additional hits per fold). We only need $c = 4$ class labels to succeed with probability 92%, *i.e.,*we need a parameter channel of 3.7 bits and have to conduct 8 trials.

In this experimental setting (parameter optimization in each of the n folds) almost arbitrarily good results are easily achievable. Ranking results achieved with this setting are distorted with high probability.

4 n-fold cross validation with fixed parameters

While in the last section the parameter optimizer was able to communicate the class labels of the first c test objects to the learner, the best the parameter optimizer can do here is to communicate the most frequently observed class label of test object i in all folds for the first c objects. If, for example, we have 3 folds and in two of them the first presented test object is of class A, the parameter optimizer may tell the learner that the first presented test object is of class A, which entails 2/3 of an extra hit averaged over the three folds.

Now what is the probability that the number of extra hits Y gained this way takes value y? We study this problem for two class labels. In this case, for each position in the test set j, $1 \leq j \leq c$, we choose the class that the majority of the n samples (drawn from position j of the n folds) belongs to. There will be $max\{y, n-y\}$ representatives of this class. Assuming that c is significantly less than m and the probability of choosing y samples is binomially distributed $B(n,p)(y)$, where p is the probability of the default class, *i.e.,*the most frequent class in the data set[2]. Hence:

$$P(Y=y) = \begin{cases} 0 & y < \frac{n}{2} \\ B(n,p)(y) + B(n,1-p)(y) & y > \frac{n}{2} \\ B(n,p)(y) & y = \frac{n}{2} \end{cases} \quad (14)$$

The total number of nz additional hits on n folds is the difference between the number of extra hits as explained in the last paragraph and the number of lost hits by not using the hypothesis h calculated in the last section:

$$P(\sum_{j=1}^{c} Y_j - \sum_{i=1}^{n} X_i \geq nz) \quad (15)$$

[2] Without this assumption the probability would be hyper-geometrically distributed, but the number of parameters needed to determine the class labels of a number of samples that gets close to the size of the test set would be outrageous.

$$= \sum_{k=nz}^{nc} P(\sum_{j=1}^{c} Y_j - \sum_{i=1}^{n} X_i = k) \quad (16)$$

$$= \sum_{k=nz}^{nc} \sum_{l=k}^{nc} P(\sum_{j=1}^{c} Y_j = l)$$

$$\cdot P(\sum_{i=1}^{n} X_i = l - k) \quad (17)$$

We can determine $P(\sum_{j=1}^{c} Y_j = l)$ and $P(\sum_{i=1}^{n} X_i = l - k)$ as follows:

$$P(\sum_{j=1}^{c} Y_j = l) \quad (18)$$

$$= \sum_{r=0}^{l} P(\sum_{j=1}^{c-1} Y_j = l - r) P(Y_c = l) \quad (19)$$

$$P(\sum_{i=1}^{n} X_i = l - k) \quad (20)$$

$$= \sum_{q=0}^{l-k} P(\sum_{i=1}^{n-1} X_i = l - k - q) P(X_n = q) \quad (21)$$

$$= \sum_{q=0}^{l-k} P(\sum_{i=1}^{n-1} X_i = l - k - q)$$

$$\cdot H(m, p_h, c)(q) \quad (22)$$

$$= \sum_{q=0}^{l-k} P(\sum_{i=1}^{n-1} X_i = l - k - q)$$

$$\cdot \frac{\binom{p_h}{q}\binom{m-p_h}{c-q}}{\binom{m}{c}} \quad (23)$$

Note that in this situation the probability of being ranked too high depends on the hit rate of a default classifier and the hit rate of the initial hypothesis: if the default hit rate is high and the initial classifier performs poorly, the probability of being over-ranked is high.

4.1 Affected benchmark problems

Diabetes: In this setting, we need $c = 7$ parameters to achieve .5 extra hits per fold with a probability of 2%. Increasing the number of parameters further decreases the probability, since the expected number of samples with equal class labels at some position in n folds, divided by the number of folds, is small compared to the hit rate of the initial hypothesis. In this experimental setting, diabetes is "safe".

Heart disease: In this data set [MST94] there are 270 samples, 2 classes, $H = .991$. Since C4.5 performs poorly for this problem we only need $c = 2$ extra hits, a parameter channel of 2 bits and 4 trials are sufficient

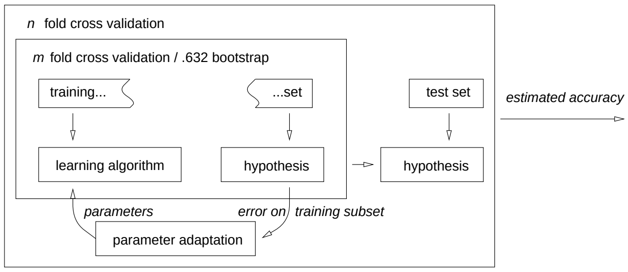

Figure 2: If model selection and accuracy estimation are separated properly, n-fold cross validation grants a bias-free estimate of the true accuracy

to succeed with > 90%. If we use k-NN as true learning algorithm (16 hits per fold), we need 6 extra hits to be ranked one rank better. With $c = 9$ samples, we succeed with probability of 90%. This is due to the fact that the default probability is not much worse than our initial hypothesis. We need a parameter channel of 8 bits and have to conduct 18 trials. We see a strong bias on this data set.

5 Unbiased assessment

In this section we want to review how unbiased ranking experiments are supposed to be conducted. What has to be kept in mind is that model selection must not be confused with accuracy estimation. The impact of this well known result on accuracy estimation of parameterized learning algorithms has to be realized. As Figure 2 illustrates, parameter adaptation needs to be performed without considering the test set. In order to get a reliable estimate of the optimal parameter settings, m-fold cross validation or .632 bootstrap should be conducted. The best parameter setting is then used to learn a classifier on the whole training set, and the hypothesis is assessed on the test set. In order to get an unbiased estimate, n-fold cross validation has to be conducted making $n \cdot m$ learning trials in total. Averaged over sufficiently many learning problems, this procedure clearly grants an unbiased estimate of the average performance of a learning algorithm, but it may be painful, since $n \cdot m$ trials need to be conducted for each problem.

6 Discussion

Our calculation clearly shows that adapting the parameters such that the accuracy on the test set is optimized causes an optimistic bias that depends on the number of parameters and the number of trials. We quantified the bias that will be observed when sufficiently many trials are conducted and the learner makes optimal use of the available parameters, and we calculated the expected number of trials needed, assuming the parameter optimizer follows a gradient descent-like search.

It shows that it is hard to be over-ranked in the one-shot training and test situation if the test set is large. In the n-fold cross validation situation, the probability of being over-ranked is high if the difference between default hit rate and true accuracy is low. If in the n-fold cross validation setting different parameter values are used (*i.e.,*the parameters are optimized locally) a highly over-ranked result can be achieved, so experiments conducted this way do not yield valid results.

Our considerations do not prove that any learning algorithm actually *is* much worse than claimed, but they do show that such claims are not validly supported by experiments in the naive setting. Our results are constructive in some sense: using the provided equations it can easily be proven that in some situations – depending on the properties of the data set – the naive but inexpensive experimental setting yields perfectly valid results.

Based on our calculations, the validation of performance evaluations for new learning algorithms seems to be difficult: many authors do not document their experimental settings to a sufficient degree, and empirical results that are based on the naive setting are likely to be distorted and cannot be compared to those obtained with unbiased experiments. Heuristic modifications that add new parameters may easily be over-estimated. Even if the modification does not improve the true accuracy of the hypothesis, a new parameter may improve the ranking results.

An important question is which additional assumptions were made in our calculation. To summarize them: we assumed that the test sets are in random order (in contrast to ordered by their class labels), but that the order is fixed, as experiments with different parameter settings are conducted. We also assumed that the parameter optimizer conducts a gradient descent for optimal parameters and showed that it will find optimal parameters if the parameters encode class labels for the first c test samples. Hill climbing algorithms for parameter adaptation are fairly common. Also, different parameter settings will result in different classification of some of the test samples, although not every distinct parameter setting may yield a different classifier. However, the correlation between parameters and class labels of test set samples is usually less direct than assumed, hence the number of experiments required to gain z extra hits may be higher, and—depending on the learner—the bias may be smaller in real world situations than in our calculation.

Parameters of learning algorithms are undesirable since the more parameters an algorithm has the less robust the algorithm is, and the harder it is to obtain a satisfactory result for a new problem. Buchanan, paraphrased by [Cat91], called this the China syndrome: Some learning algorithms have so many parameters that the only person who can make the program run is with high probability currently in China. Here we have illustrated the following extension of this syndrome, which might be considered Murphy's law on machine learning algorithms: the algorithm that is ranked most highly, and is therefore considered most suitable for a given problem, is the one that is least likely to run and grant a satisfactory result because it has so many parameters that the only person who claims to be able to make it run is, with high probability, currently unavailable.

Acknowledgment

This work was supported by a grant of the DFG and an Ernst-von-Siemens fellowship. We wish to thank Prof. Alberto Segre who did a great job of helping us to edit this paper, and Prof. Wysotzki for the interesting discussion.

References

[Cat91] J. Catlett. *Megainduction: machine learning on very large data bases.* PhD thesis, University of Sydney, 1991.

[DM94] B. Dolsak and S. Muggleton. The application of inductive logic programming to finite-element mesh design. In S. Muggleton, editor, *Inductive Logic Programming*, pages 453–472. Academic Press, 1994.

[Efr79] B. Efron. Bootstrap methods: another look at the jacknife. *Annals of Statistics*, 7(1):1–26, 1979.

[Efr83] B. Efron. Estimating the error rate of a prediction rule. *Journal of the American Statistical Association*, 68(382):316–330, 1983.

[KJ97] R. Kohavi and G. John. Wrappers for feature subset selection. *J. AI, Special Issue on Relevance*, 1997. to appear.

[Koh95] R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs.* PhD thesis, Stanford University, 1995.

[MST94] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[Qui90] J. R. Quinlan. Learning logical definitions from releations. *Machine Learning*, 5:239–266, 1990.

[Qui92] J. R. Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann Publisher, 1992.

[Ren70] A. Renyi. *Probability Theory.* North-Holland, 1970.

[Sto74] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, B 36:111–147, 1974.

[Zha92] J. Zhang. On the distributional properties of model selection criteria. *Journal of the American Statistical Association*, 87(419):732–737, 1992.