




# Bagging using Statistical Queries

**Abstract.** Bagging is an ensemble method that relies on random resampling of a data set to construct models for the ensemble. When only statistics about the data are available, but no individual examples, the straightforward resampling procedure cannot be implemented. The question is then whether bagging can somehow be simulated. In this paper we propose a method that, instead of computing certain heuristics (such as information gain) from a resampled version of the data, estimates the probability distribution of these heuristics under random resampling, and then samples from this distribution. The resulting method is not entirely equivalent to bagging because it ignores certain dependencies among statistics. Nevertheless, experiments show that this “simulated bagging” yields similar accuracy as bagging, while being as efficient and more generally applicable.

## 1 Introduction

Ensemble methods build a set of different models and combine their predictions to classify new examples. These methods vary in the way they build the separate models and in the way they combine their predictions. Bagging [1] builds several models on replicate training sets that are produced by sampling with replacement from the original training set. By doing so it is able to improve the predictive accuracy.

In some learning settings, direct access to individual examples of the data is not available; instead, the learning algorithm has access only to summary statistics about the data. Several authors have described how various predictive models (such as decision trees) can be learned from precomputed statistics such as itemset frequencies [2] or AD-trees [3], and what the advantages are of doing so. When trying to apply bagging in the context of learning from statistics, one is confronted with the fact that statistics are needed for several resampled versions of the dataset, rather than for the dataset itself. Since no direct access to the data is available, resampling the dataset by randomly selecting (with replacement) individual examples from it is not possible, so the straightforward approach of resampling the dataset and computing the statistics for these resampled versions is not possible. An alternative is to simulate the computation of statistics from randomly resampled versions of the dataset by computing the distribution (under random resampling of the dataset) of the statistics themselves,

and then sampling values for these statistics from those distributions. This approach, applied to the specific case of bagging decision trees, is the subject of this paper.

The paper is organized as follows. First we describe the learning problem in Sect. 2. Then Sect. 3 deals with bagging: we explain the basic algorithm we use in the bagging procedure, namely decision trees, and show how it is applied using statistical queries. Next we briefly focus on bagging. In Sect. 4 we provide a new way to simulate the use of bootstraps and indicate how this method differs from the original bagging procedure. Section 5 describes experimental results comparing this new method to the original bagging procedure. These results show that bagging can be simulated efficiently with the proposed approach. In Sect. 6 we conclude and provide some directions for future work.

## 2 Learning from Statistics

We start with defining and motivating the exact problem setting. We address the problem of classification using only statistical information about the data set to learn a classification model. We assume the data set is described by  $m$  attributes ( $A_1 \dots A_m$ ) from which  $A_m$  is the class attribute  $C$  consisting of  $n_c$  possible class labels ( $c_1 \dots c_{n_c}$ ). The learning algorithm does not have access to the training set  $E$  but is provided an oracle  $O$  that can be queried for frequencies  $fr(Cr) = |\{e \in E | Cr(e)\}|$ , where  $Cr \in \mathcal{C}$  is taken from some (possibly restricted) space of conditions. For instance,  $\mathcal{C}$  might contain all conditions of the form  $A_i = a$  with  $a$  some value of  $A_i$ ; all conjunctions of such conditions; all such conjunctions with at most  $K$  conjuncts; all such conjunctions of which the frequency is above some predefined threshold; etc. Note that if there are no restrictions to  $\mathcal{C}$  (more specifically, if  $\mathcal{C}$  is sufficiently expressive to identify any individual instance), then full information on the dataset is available and then this setting does not differ essentially from learning from the dataset itself, except for implementation issues.

Several researchers have worked in this setting. For instance, after running an algorithm such as Apriori [4] on a dataset, the frequencies of all itemsets with a frequency above Apriori's minimal support parameter have been computed and stored; Panov et al. [2] show how predictive models such as decision trees, decision rules and Naive Bayes can be computed from only this information. Similarly, Moore and Lee [3] proposed AD-trees, a data structure to efficiently store the frequency of any conjunction of attribute-value combinations in a dataset, and showed how for instance decision trees can be learned from AD-trees.

The advantage of this setting is that, once an efficient oracle is available (the AD-tree has been built, or the itemsets mined), predictive models can be built with far greater speed than when having to recompute all the necessary statistics (which involves at least one pass over the whole dataset) each time they are needed. The setting is also useful when mining databases where for reasons of privacy only statistical queries are allowed.

The oracle knows certain statistics about a single data set, the training set. But some learners employ randomly resampled versions of the training set. For instance, as Panov et al. [2] mention, Ripper prunes its rule sets by means of random resampling. Also bagging uses resampled versions of the training set. It is not obvious how such methods can be used in the setting of learning from frequencies. For bagging, we investigate this in this paper.

### 3 Bagging

#### 3.1 Decision Trees

Decision trees are usually constructed top-down, from the root to the leaves. At each node of the tree a “most informative” test for that node is selected using some heuristic. Different heuristics have been proposed to select the best test [5, 6]. In the remainder of this text we use information gain, but our method also works for other heuristics. The information gain  $\mathcal{IG}_A(E)$  of an attribute  $A$  relative to a set of examples  $E$  is defined as  $\mathcal{IG}_A(E) = \mathcal{E}(E) - \sum_{v \in \text{Values}(A)} \frac{|E_v|}{|E|} \mathcal{E}(E_v)$  where  $\mathcal{E}(E)$  is the entropy of  $E$  ( $\mathcal{E}(E) = \sum_{i=1}^c p_i \log_2 p_i$ , where  $p_i$  is the proportion of  $E$  belonging to class  $i$ ). We see that the heuristic is determined by the class distributions  $p_i$  within a node, which in turn directly follow from frequencies  $fr(c \wedge L)/fr(L)$  where  $L$  represents the conditions on the path from the tree’s root to the current node,  $fr(L)$  is the number of examples covered by this node, and  $fr(c \wedge L)$  is the number of such examples of class  $c$ . Similarly, other statistics used by tree learners (e.g., for the stopping criterion) can be defined in terms of such frequencies. Panov et al. [2] describe how a good decision tree can be learned even from incomplete statistics (when the frequency of itemsets is unknown if it is below a certain threshold). In this paper we assume that decision trees can be learned in the described setting, and we focus on the bagging procedure.

#### 3.2 Bagging

Bagging [1] operates by repeatedly resampling the training set and building trees on these resamples. The resamples  $E_i$  form replicate data sets (also called bootstraps), each consisting of  $n$  examples (with  $n$  equal to the size of the original data set  $E$ ), drawn at random, but with replacement, from  $E$ . So each example  $e$  from  $E$  may appear repeated times or not at all in any particular  $E_i$ .

But since in our setting, examples cannot be accessed individually, straightforward sampling with replacement on the original data cannot be performed anymore. We only have statistics about the original data available. In the next section we explain how we will sample the statistics instead of the dataset.

### 4 Sampling the Statistics

The key idea to bagging without bootstrapping is the following. In classical bagging, the information gain  $\mathcal{IG}_A$  (or some other heuristic) of an attribute  $A$  is

computed from a random resample  $E_i$  of the original data set  $E$ . Since the resample is chosen randomly, it might just as well have been another one, which would have led to a different value for  $\mathcal{IG}_A$ . Clearly, computing the exact  $\mathcal{IG}_A$  on a random resample  $E_i$  is equivalent to sampling  $\mathcal{IG}_A$  from its own distribution. Put differently: since  $\mathcal{IG}_A$  is a function of the data set, and considering  $E_i$  a stochastic variable of which the distribution is known, the distribution of  $\mathcal{IG}_A(E_i)$  can be computed. Generating values for  $\mathcal{IG}_A$  from this distribution is equivalent to generating  $E_i$  from its own distribution and computing  $\mathcal{IG}_A$  from it. We will refer to this procedure as “resampling the statistics” as opposed to “resampling the data sets” (and computing the statistics from them).

In our approach, we will not resample information gain directly; the statistics that we resample are frequencies of class and attribute values, from which information gain but also other useful statistics can be computed efficiently.

**Resampling Statistics for a Single Test.** Assume for the sake of illustration that we have 3 boolean attributes (A, B, C) and a binary class variable with values +, -. Figure 1a shows for each attribute the joint distribution of that attribute and the class attribute in a training set with 100 examples. (Note that in this paper the term distribution will usually refer to a sample distribution, measured by absolute frequencies, and not a population distribution.) From the frequencies shown there, the information gain of A, B and C can easily be computed. If we take a bootstrap sample of 100 instances from the training set, what will the joint distribution of the attributes and the class variable look like? Each instance, being randomly taken from the training set, has a  $60/100 = 0.6$  chance of being (+, -A), a 0.2 probability of being (-, A), a 0.5 probability of being (+, B), etc., and hence has the same probability of ending up in the corresponding cell in the joint distribution table of the resampled set. Clearly, the vector  $(n_{+,A}, n_{+,-A}, n_{-,A}, n_{-,-A})$  is multinomially distributed with parameter  $(0, 0.6, 0.2, 0.2)$ , and similarly for B and C.

Generally, if we have  $n_c$  classes  $c_1, \dots, c_{n_c}$  and  $n_a$  values  $a_1, \dots, a_{n_a}$  of an attribute  $A$ , then the  $n_a n_c$ -dimensional vector  $(X_{11}, \dots, X_{n_a n_c})$  where  $X_{ij}$  denotes the number of instances in dataset  $E$  with  $A = a_i$  and class value  $c_j$ , is multinomially distributed with parameters  $(p_{11}, \dots, p_{n_a n_c})$  where  $p_{ij} = X_{ij}/n$ , with  $n = \sum X_{ij}$  (the total number of instances in  $E$ ):

$$P(X'_{11} = x_1, \dots, X'_{n_a n_c} = x_k) = \frac{n!}{x_1! \dots x_k!} p_{11}^{x_1} \dots p_{n_a n_c}^{x_k}$$

where  $k = n_c * n_a$ . We use the method proposed by Devroye [7] to generate this vector; this method consists of repeatedly generating a number for each separate component  $X_{ij}$  (so  $n_a * n_c$  times) according to a binomial distribution, using the BTPE algorithm from [8].

**Choosing the Best Test for a Single Node.** With the above method for generating the  $X'_{ij}$ , we can accurately model how the  $\mathcal{IG}$  of each attribute will be

	$A$	$\neg A$		$B$	$\neg B$		$C$	$\neg C$
+	0	60	+	50	10	+	30	30
-	20	20	-	10	30	-	30	10

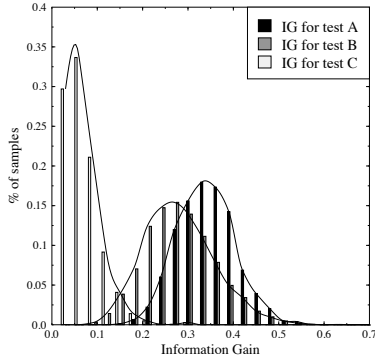
(a) Example of statistics  $S[t]$

	$ABC$	$A\neg BC$	$AB\neg C$	$A\neg B\neg C$
+	0	0	0	0
-	5	0	12	3

	$\neg ABC$	$\neg A\neg BC$	$\neg AB\neg C$	$\neg A\neg B\neg C$
+	30	0	20	10
-	3	10	2	5

(b) Example of a joint distribution of Fig. 1a



(c) The distribution of information gains on random resamples of the original data for test A, B and C according to Fig. 1a

**Fig. 1.**

distributed, under the resampling conditions used by bagging. Figure 1c shows the distribution of the  $\mathcal{IG}$ s of attributes A, B and C under bootstrap resampling of the training set summarized in Fig. 1a. It can be seen, for instance, that the probability that C is the best test in any random bootstrap is very low.

Unfortunately, the situation is not so clear for A and B. It might seem that we get a good estimate of the probability that  $\mathcal{IG}_A > \mathcal{IG}_B$  by just generating random  $\mathcal{IG}_A$  and  $\mathcal{IG}_B$  and checking how often  $\mathcal{IG}_A > \mathcal{IG}_B$ . But with this procedure we are assuming that  $\mathcal{IG}_A$  and  $\mathcal{IG}_B$  are independent. This is not necessarily the case: A and B may be correlated and so may their  $\mathcal{IG}$ s. Consider for instance the joint distribution over A, B, C and the class, shown in Fig. 1b. This distribution gives the same marginal distributions as shown in Fig. 1a and in Fig. 1c. Yet, this joint distribution is such that  $P(\mathcal{IG}_A > \mathcal{IG}_B)$  is approximately 1, whereas sampling  $\mathcal{IG}_A$  and  $\mathcal{IG}_B$  independently gives  $P(\mathcal{IG}_A > \mathcal{IG}_B) = 0.72$ .

It is impossible to correctly compute  $P(\mathcal{IG}_A > \mathcal{IG}_B)$  from the distributions of  $\mathcal{IG}_A$  and  $\mathcal{IG}_B$ ; we need the joint distribution. To compute the latter we need the joint frequency distribution of A, B and the class. Generally, to correctly mimic bagging, the full joint distribution of all attributes and the class is required.

Unfortunately, using the full joint distribution may be infeasible in practice. First of all, knowledge of the full distribution corresponds to perfect knowledge of the training dataset; it may not really be interesting to try to simulate bagging in that context (unless, for instance, it would yield an efficiency gain). Further, the number of  $X'$  variables that need to be generated grows exponentially in the number of attributes, which makes this approach practically infeasible.

Of course, while we know that our approach is theoretically only an approximation of what happens in bagging, the question remains how much this matters in practice. In practical datasets, the correlations between attributes may be such that the probability of an attribute having the highest information gain under

our approximate method or true bagging seldom differs strongly; and if these probabilities differ, there is still the question of how much this affects the quality of the bagged ensemble. The latter question is of more practical importance and we will focus on that in our experiments.

**Choosing the Best Test in Multiple Nodes.** The procedure described above assumes we are in the root node of the tree. Deeper in the tree, the statistics we resample for each possible test  $A_i$  in a certain node with path  $L$  from the root are the frequencies  $fr(L \wedge A_i = a \wedge C = c)$  for each value  $a$  of  $A_i$  and each value  $c$  of the class attribute  $C$ . This means that for the different paths in the tree we need joint distributions over several attributes, but this is usually only a small subset of the full joint distribution. Note that while in real bagging one complete tree is built on one particular bootstrap, here new samples are drawn at each node. Again, we see that a perfect simulation of bagging is only possible by using the full joint distribution, which may not be feasible. So it remains to be seen experimentally how much our simulated bagging procedure differs from actual bagging.

## 5 Experiments

We implemented this simulated bagging method in the WEKA data mining system [9]. As a basic decision tree algorithm we started from ID3 [10]. For the binomial random number generation we used the CERN Java library [11] containing a high performance implementation of the BTPE algorithm [8]. More details about the implementation can be found in [12].

We compared the accuracy of simulated bagging (using resampling of the statistics) to original bagging (using resampling the data) on 12 UCI data sets<sup>1</sup>. We selected data sets with a varying data set size (from 286 to 67557 examples), all consisting of mainly nominal attributes, as this is required by the ID3 decision tree algorithm. We performed discretization where still necessary and removed missing values. To allow accuracy comparison between simulated bagging and the original bagging approach, no constraints were imposed on the frequencies that simulated bagging could query. Both methods are performed using 20 iterations of bagging and are evaluated by averaging over 5 stratified ten-fold cross-validations. Results on accuracy, size of the ensembles and runtimes are shown in Table 1 for simulated bagging (SB), simulated bagging with an adapted stopping criterion (SB\_stop, discussed below) and regular bagging (Bag). As can be seen from the upper part of the table, in general the accuracy results are not significantly different from each other (also when taking standard deviations into account). Also, despite the fact that simulated bagging neglects dependencies among attributes, we found that the number of different tests chosen among the different iterations at a certain level in the trees was very similar for the two

<sup>1</sup> breast-cancer (br), kr-vs-kp (kr), primary-tumor (pr), soybean (so), splice (sp), waveform-5000 (wa), credit-a (cr), hypothyroid (hy), mushroom (mu), vote (vo), nursery (nu), connect-4 (co)

**Table 1.** Results on accuracy, size of the ensembles and runtime of simulated bagging (SB), SB with stopping criterion (SB\_stop) and normal bagging (Bag) on 12 UCI data sets (abbreviations of data sets are mentioned in text above).

	br	kr	pr	so	sp	wa	cr	hy	mu	vo	nu	co
	Accuracy											
SB	65.7	99.6	39.8	91.7	94.3	79.1	82.7	99.5	100.0	95.3	99.1	79.6
SB_stop	65.9	99.6	39.8	91.7	94.3	79.6	84.2	99.5	100.0	95.7	99.0	80.0
Bag	66.0	99.6	36.4	90.1	93.8	77.4	82.6	99.4	100.0	94.6	99.0	79.3
	Size (number of nodes)											
SB	7262	1931	7287	4351	19941	100863	8346	1933	575	1400	17471	968167
SB_stop	4919	1720	5231	3155	14127	66582	5970	1592	566	1089	13432	671063
Bag	4523	1302	4980	2545	8416	42042	4276	1113	502	732	16006	534225
	Time (seconds)											
SB	0.436	1.102	1.616	1.050	6.396	6.466	0.556	0.440	0.252	0.214	1.156	209.786
SB_stop	0.296	0.986	1.178	0.788	5.140	4.154	0.404	0.400	0.246	0.180	0.946	100.384
Bag	0.282	2.216	0.842	1.078	5.692	5.574	0.566	1.298	2.060	0.224	3.074	187.624

methods. The only point where trees of the two methods really differ is in their size. The middle part of Table 1 shows that sizes of the ensembles output by simulated bagging are in general larger than those of the ensembles of regular bagging. This is because in simulated bagging the frequencies that are queried in all nodes are always computed on the complete data set and not on a bootstrap, which only contains 63% of the original examples. Table 1 also shows results when applying a stopping criterion that only looks at 63% of the data while checking the minimal leaf size condition to decide whether to stop or not, which results in shorter trees.

From the efficiency results, we can see that even in a normal learning setting where the data set is provided and regular bagging can be applied, simulated bagging can be useful as it yields similar accuracy often in less time. In [12] we discuss the time complexity of using this simulated bagging in a normal learning setting where the data set is available and compare it to regular bagging and a more efficient implementation of bagging [13]. We point out conditions under which simulated bagging might be preferred over regular bagging.

We conclude that ignoring dependencies between tests when sampling the statistics does not have a detrimental effect, while making bagging more generally applicable.

## 6 Conclusions

We have proposed a new method for approximating bagging in a learning setting where only statistics about the data and not the individual data instances themselves are available. In this context resampling the data set (with replacement), as is usually done by bagging, cannot be applied anymore. Here we propose a simulation of bagging that resamples the statistics instead of the data. Although it ignores certain dependencies among the statistics, it shows to be a good and efficient approximation to bagging.

The applicability of the described technique is not restricted to bagging. It can be applied to all methods using resampling, such as certain pruning methods (Panov et al.[2] described the need for that), some methods for error assessment, etc.

Our current implementation, based on ID3 [10], only handles data sets with nominal attributes. For numerical attributes the basic ideas presented here still apply, but there are some technical details that are more complicated. When computing statistics for each possible threshold for a numeric attribute, it suffices to go over the data set only once if examples are sorted according to the attribute. Then for each of the iterations of bagging we could take a sample of each of these statistics and compute the information gains on them. But statistics of successive thresholds are strongly correlated and by randomly sampling we would lose this characteristic. So after we have taken a sample for the first threshold we might want to impose some constraints on the samples of the next thresholds. This method will be investigated in the near future.

### Acknowledgements

We would like to thank the reviewers and chairs for their valuable comments and suggestions.

### References

1. Breiman, L.: Bagging predictors. *Machine Learning* **24**(2) (1996) 123–140
2. Panov, P., Džeroski, S., Blockeel, H., Loškowska, S.: Predictive data mining using itemset frequencies. In: Proc. of the 7th Int. Multiconf. Information Society. (2005)
3. Moore, A.W., Lee, M.S.: Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research* **8** (1998) 67–91
4. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In Buneman, P., Jajodia, S., eds.: Proc. of ACM SIGMOD Conf. on Management of Data, Washington, D.C., USA, ACM (1993) 207–216
5. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann (1993)
6. Breiman, L., Friedman, J., Olshen, R., Stone, C.: *Classification and Regression Trees*. Wadsworth, Belmont (1984)
7. Devroye, L.: *Non-Uniform Random Variate Generation*. Springer-Verlag (1986)
8. Kachitvichyanukul, V., Schmeiser, B.: Binomial random variate generation. *Communications of the ACM* **31** (1988) 216–222
9. Witten, I., Frank, E.: *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann (2005) 2nd Edition.
10. Quinlan, J.R.: Induction of decision trees. *Machine Learning* **1** (1986) 81–106
11. Cern: European org. for nuclear research (1999) <http://dsd.lbl.gov/~hoschek/colt/>.
12. Van Assche, A., Blockeel, H.: Simulating bagging without bootstrapping. In Saeys, Y., Tsiporkova, E., De Baets, B., Van de Peer, Y., eds.: Proc. of the 15th Annual Machine Learning Conf. of Belgium and the Netherlands. (2006) 25–32
13. Blockeel, H., Struyf, J.: Efficient algorithms for decision tree cross-validation. *Journal of Machine Learning Research* **3**(Dec) (2002) 621–650