Query Learning Strategies using Boosting and Bagging

Naoki Abe **Hiroshi Mamitsuka**
Theory NEC Laboratory, RWCP*
c/o NEC C& C Media Research Laboratories
4-1-1 Miyazaki, Miyamae-ku, Kawasaki 216-8555 JAPAN
{abe, mami}@ccm.cl.nec.co.jp

Abstract

We propose new query learning strategies by combining the idea of query by committee and that of boosting [Sch90, FS95] and bagging [Bre94]. Query by committee is a query learning strategy which makes use of a randomized component learning algorithm and works by querying the function value of a point at which the predictions made by many copies of the component algorithm are maximally spread. The requirement of query by committee on the component algorithm that it be an ideal randomized algorithm makes it hard to apply in practice when we have only a moderately performing deterministic algorithm. To address this issue, we borrow the ideas of boosting and bagging, which are both techniques to enhance the performance of an existing learning algorithm by running it many times on a set of re-sampled data and combining the output hypotheses to make a prediction by (weighted) majority voting. We propose two query learning methods, query by bagging and query by boosting, which select the next query point by picking a point on which the (weighted) majority voting by the obtained hypotheses has the least margin. We empirically evaluate the performance of these methods on a wide range of real world data. Our experiments show that, when using C4.5 as the component learning algorithm and run on data sets in UCI Machine Learning repository, both query learning methods significantly improve data efficiency as compared to both C4.5 itself and boosting applied on C4.5. A typical increase in data efficiency achieved was 2 to 4-fold.

*Real World Computing Partnership

1 Introduction

Query learning is a sub-area of machine learning attracting increasing attention both in theory and in practice with the expectation that it may bring down both computational and sample complexities that plague passive learners. (c.f. [LC94, CS94, LG94]) For example, there is a rich body of work on the algorithmic approach to query learning as initiated by Angluin's query learning model [Ang87]. Another promising approach is the Bayesian or information theoretic approach to query learning [PK95, SOS92], in which a query learner tries to maximize the information gain on each query. Of the latter approach, 'query by committee' [SOS92] is an especially attractive and general query learning strategy with theoretical performance guarantee. In the present paper, we propose new variants of query by committee, which we call 'query by boosting' and 'query by bagging,' by combining query by committee with the techniques of boosting and bagging.

'Query by committee' [SOS92] is a query learning strategy which makes use of many copies of an ideal randomized learning algorithm. More concretely, it uses a number of copies of Gibbs algorithm (a randomized algorithm that picks a hypothesis from a given hypothesis class according to the posterior distribution and predicts according to it) and queries the function value of a point at which their predictions are maximally spread. The idea is that, by choosing a query point with maximum uncertainty of estimation of its function value, the information gain can be maximized. Indeed, there is a theoretical guarantee of the near-optimality of the data efficiency of this method, but it is based on the assumption that the component learning algorithm is Gibbs algorithm. This assumption poses two problems when one tries to apply this technique in practice: One is the problem of computational complexity, because Gibbs algorithms for interesting hypothesis classes tend to be computationally intractable. The other is that it cannot be applied on a deterministic component learning algorithm. The two methods we propose in the present paper, 'query by boosting' and 'query by bagging,' are motivated to address these two issues.

'Boosting' and 'bagging' are both techniques to enhance the performance of an existing learning algorithm by running it many times on a set of re-sampled data and combining the output hypotheses to make a prediction. Bagging, due to Brieman [Bre94], is the simpler of the two, and it works by re-sampling from the input data with the same (uniform) distribution and its final hypothesis is obtained by taking majority vote over the predictions of the output hypotheses. Boosting[1] [Sch90, FS95] is a more complicated method that can be used to boost the performance of a relatively weak learning algorithm by use of sophisticated re-sampling on the training data. It does so by repeatedly re-sampling on the input training data, with the sampling distribution varied each time so as to focus more and more on the part of the training data on which the previously obtained hypotheses did poorly on. The final prediction of boosting is made by taking a weighted majority (or average) of the predictions of all the hypotheses thus obtained.

As noted earlier, one of the weakness of query by committee is that it cannot be applied on a deterministic component algorithm. If the component learning algorithm we have available is deterministic, the idea of bagging offers a natural alternative; namely apply bagging to obtain a set of hypotheses, let these hypotheses predict on a set of candidate points, and pick the point on which the predictions have the largest variance. When making a prediction, predict by majority vote over all the hypotheses. Since query by bagging introduces randomness in the form of re-sampling from the input data, it can be used on a component algorithm that is deterministic.

When the learning problem of interest is sufficiently complex, efficient implementation of Gibbs algorithm is not possible. If such is the case and the best known learning algorithm does not have a very good performance, then it makes sense to use boosting to enhance its performance. Recall that the most notable characteristic of boosting is its tolerance on the performance of the component learning algorithm. Thus, appropriately combining the idea of boosting and query by committee, we may obtain a query learning method that is tolerant on the performance of the component learning algorithm.

Recent experimentation using boosting has shown a remarkable fact (e.g. [DSS92]) that even after boosting has achieved perfect prediction on the training data, it keeps boosting its predictive performance on unseen data. This seemingly contradicts known facts about over-learning, but recently Schapire *et al* [SFBL97] have given an account of this fact. That is, even after realizing perfect predictive performance on the training data, boosting keeps increasing its confidence of prediction, or more specifically the difference between the total weight assigned to the correct prediction and that assigned to a wrong prediction. (This is called the 'margin' of the prediction.) In their paper, they prove that a hypothesis having a larger margin on the training data performs better on unseen data as well. Based on this observation, the method we propose here, query by boosting, selects as the next query a point on which the margin obtained by the boosting algorithm is minimum, and attempts to maximize the uncertainty of prediction and hence the information gain on each query.

We conducted experiments using real world data to evaluate the performance of the proposed query learning methods. In particular, we tested them on a large part of the UCI Machine Learning data repository, using Quinlan's C4.5 as the component algorithm. Here we note that testing query learning algorithms on these databases is not possible in a strict sense, since not all the query points can be answered. We therefore used our query strategies as methods of *selective sampling* to pick more informative queries from a fixed set of training data. (c.f. [LG94]) On almost all the data sets we tested these learning methods, both query by boosting and query by bagging achieved significant increase in data efficiency as compared to both C4.5 and boosting applied on C4.5. The increase in data efficiency measured by the data size required by the query learning methods to reach the same accuracy achieved by C4.5 (near the end of the data set) was anywhere from 2 to 5-fold. As compared to boosting applied on C4.5, the increase in data efficiency of the query methods was 2 to 4-fold on most data sets.

On one of the eight data sets above, tic-tac-toe, we ran analogous experiments using a different component learning algorithm – a randomized version of a weighted majority prediction algorithm for learning n-ary relations proposed in [ALN95] called WMP1. In addition to the two query methods, we also tested the original query by committee method, as the component algorithm is now randomized. It was found that, with randomized WMP1 as the component algorithm, both query by boosting and query by bagging performed better than query by committee.

[1]Boosting was first discovered by Schapire [Sch90] in the context of proving the equivalence of 'weak learnability' with the strong PAC learnability. It was subsequently improved by Freund [Fre90], and Freund and Schapire [FS95].

```
**Algorithm: Query-by-Committee(QBC)**
Notation: In general, we use $S'_i$ to denote the
 unlabeled sample corresponding to $S$.
Input: Number of trials: $N$
 Randomized component learning algorithm: $A$
 Number of times $A$ is called: $T$
 Number of query candidates: $R$
 A set of query points: $Q$
Initialization: $S_1 = \langle x_1, f(x_1)\rangle$ for random $x_1$
For $i = 1, ..., N$
 1. Run $A$ on $S_i$ $T$ times to obtain $h_1, ..., h_T$.
 2. Randomly generate a set of $R$ points $C \subset Q \setminus S'_i$
  with respect to uniform distribution over $Q \setminus S'_i$.
 3. Pick a point $x^* \in C$ split most evenly: $x^* = \arg$
 $\min_{x \in C} ||\{t \leq T | h_t(x) = 1\}| - |\{t \leq T | h_t(x) = 0\}||$
 4. Query the function value at $x^*$ and obtain $f(x^*)$.
 5. Update the past data as follows
  $S_{i+1} = append(S_i, \langle x^*, f(x^*)\rangle)$
End For
Output: Output as the final hypothesis:
  $h_{fin}(x) = \arg\max_{y \in Y} |\{t \leq T | h_t(x) = y\}|$
  where $h_t$ are hypotheses of the final ($N$-th) stage
```

Figure 1: Query by Committee (QBC)

```
Algorithm: Query-by-Bagging(QBag)
Input: Number of trials: $N$
 Component learning algorithm: $A$
 Number of times re-sampling is done: $T$
 Number of query candidates: $R$
 A set of query points: $Q$
Initialization: $S_1 = \langle x_1, f(x_1)\rangle$ for random $x_1$
For $i = 1, ..., N$
 1. By resampling according to uniform distribution
  on $S_i$, obtain sub-samples $S'_1, .., S'_T$ each of size $m$.
 2. Run $A$ on each sub-sample and obtain $h_1, ..., h_T$.
 3. Randomly generate a set of $R$ points $C \subset Q \setminus S'_i$.
  with respect to uniform distribution over $Q \setminus S'_i$.
 4. Pick a point $x^* \in C$ split most evenly: $x^* = \arg$
 $\min_{x \in C} ||\{t \leq T | h_t(x) = 1\}| - |\{t \leq T | h_t(x) = 0\}||$
 5. Query the function value at $x^*$ and obtain $f(x^*)$.
 6. Update the past data as follows
  $S_{i+1} = append(S_i, \langle x^*, f(x^*)\rangle)$
End For
Output: Output as the final hypothesis:
  $h_{fin}(x) = \arg\max_{y \in Y} |\{t \leq T | h_t(x) = y\}|$
  where $h_t$ are hypotheses of the final ($N$-th) stage
```

Figure 2: Query by Bagging (QBag)

2 Query Learning Methods

2.1 Query by Committee

We briefly describe the original query by committee method, generalized to use an arbitrary randomized component algorithm. At any point in time, query by committee runs the component algorithm on the past data a number of times to obtain many hypotheses. It picks the next query point by choosing from among a set of randomly generated candidate points a point such that the predictions by the hypotheses are split most evenly. The details are given in Figure 1. Here, if Q is a pre-determined set of points on which the function values can be obtained, then the algorithm as described is a method of selective sampling. If, on the other hand, Q is set to the entire domain, then it is a genuine query learning algorithm, which is free to choose any point in the domain as a query point.

2.2 Query by Bagging

'Bagging'[Bre94] re-samples from the input sample with a fixed distribution, and the final hypothesis is obtained by averaging the outputs of the hypotheses thus obtained. This method is based on the idea that prediction error consists of the 'bias,' which is the estimation error necessitated by the input data size, and the 'variance' which is due to the statistical variation existing in the specific data. The claim is that bagging can isolate the two factors and can minimize the variance component of the error. Query by bagging is like query by committee, except it applies bagging on the input sample and picks as the next query point a point at which the predictions of the hypotheses are most evenly split. The details of query by bagging are also given in Figure 2.

2.3 Query by Boosting

We will now describe the query by boosting method in detail. In query by boosting, we pick as the next query point a point at which the weighted voting of the final hypothesis obtained by boosting the component learning algorithm has the least 'margin.' When the target function is 0,1-valued, this means that the query point is one for which the difference between the total weight for the value 1 and that for 0 is minimum among all candidate points. We give the details of this procedure in Figure 3, where we also supply the details of AdaBoost [FS95] for completeness.

Note that the original query by committee, query by bagging, and query by boosting form a natural progression. In query by committee, all the samples are *identical*, and the variance of the component algorithm's predictions is taken with respect to the randomness that exists within the component algorithm. In query by bagging, subsamples are obtained from the input sample using an *identical* distribution, and the variance of the component algorithm's predictions is with respect to the randomness in re-sampling. In

```
Algorithm: Query-by-Boosting(QBoost)
Input: Number of trials: $N$
  Component learning algorithm: $A$
  Number of times re-sampling is done: $T$
  Number of query candidates: $R$
  A set of query points: $Q$
Initialization: $S_1 = \langle x_1, f(x_1) \rangle$ for random $x_1$
For $i = 1, ..., N$
  1. Run AdaBoost on input ($S_i$, $A$, $T$) and get:
     $h_{fin}(x) = \arg\max_{y \in Y} \sum_{h_t(x)=y} \log \frac{1}{\beta_t}$
  2. Randomly generate a set of $R$ points $C \subset Q \setminus S_i'$.
     with respect to uniform distribution over $Q \setminus S_i'$.
  3. Pick a point $x^* \in C$ with the minimum margin:
     $x^* = \arg\min_{x \in C} |\sum_{h_t(x)=0} \log \frac{1}{\beta_t} - \sum_{h_t(x)=1} \log \frac{1}{\beta_t}|$
  4. Query the function value at $x^*$ and obtain $f(x^*)$.
  5. Update the past data as follows
     $S_{i+1} = append(S_i, \langle x^*, f(x^*) \rangle)$
End For
Output: Output $h_{fin}$ in the last stage as the output.
```

```
Subroutine: AdaBoost [FS95]
Input: Sample: $S = \langle (x_1, y_1), .., (x_i, y_i), .., (x_m, y_m) \rangle$
  (Here, assume $\forall y_i \in Y = \{0, 1\}$.)
  Component learning algorithm: $A$
  Number of times re-sampling is done: $T$
Initialization: $\forall i \leq m, D_1(x_i) = \frac{1}{m}$
For $t = 1, .., T$
  1. Run $A$ on a sample of size $m$ generated w.r.t. $D_t$.
  2. Let its output hypothesis be $h_t$.
  3. Compute its error rate $\epsilon_t$ by:
     $\epsilon_t = \sum_{h_t(x_i) \neq y_i} D_t(x_i)$
  4. Calculate $\beta_t$ by $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
  5. Update the re-sampling distribution $D_{t+1}$:
     $D_{t+1}(x_i) = \frac{D_t(x_i) \cdot \beta_t}{Z}$  if  $h_t(x_i) = y_i$
     $D_{t+1}(x_i) = D_t(x_i)$  otherwise
  (Here $Z$ is a normalization constant satisfying
  $\sum_{i=1,..,m} D_{t+1}(x_i) = 1$.)
Output: Output as the final hypothesis:
  $h_{fin}(x) = \arg\max_{y \in Y} \sum_{h_t(x)=y} \log \frac{1}{\beta_t}$
```

Figure 3: Query by boosting (QBoost)

name	# ex.	# attributes disc.	cont.	missing values
liver-disorders	345	-	6	-
ionosphere	351	-	34	-
house-votes-84	435	16	-	o
wdbc	569	-	32	-
crx	690	9	6	o
breast-cancer-wisconsin	699	9	-	o
pima-indians-diabetes	768	-	8	-
tic-tac-toe	958	9	-	-

Table 1: The eight data sets used in our experiments.

query by boosting, the re-sampling distribution itself is changed depending on the properties of the obtained hypotheses, and the variance of the component algorithm's predictions is measured with respect to the uncertainty involved in weighted voting by the various hypotheses.

3 Experimental procedures

We evaluate the proposed query learning methods on the learning problem for concepts (or 0,1-valued functions) over a number of attributes, which are either binary, discrete or numerical. A special case of this is when all the attributes are discrete, and the target function can be regarded as an n-ary relation over n finite sets. In our experiments, we use existing data sets for training and test data, without an explicitly defined target function. Since it is not possible to use query learning algorithms genuinely as query learners in this setting, we use them as methods for *selective sampling*, that is, ways to select a smaller set of more effective data from a large data set.

The data sets we used in our experiments were borrowed from the machine learning data repository of University of California at Irvine.[2] Of the large number of data sets available from the repository, we selected 8 (not all) data sets satisfying the following conditions: (1) The target function is 0,1-valued; (2) The data size is moderate (more than 300 and less than 1,000); Table 1 summarizes the data sets we selected and their basic characteristics.

On these data sets, we compared the performance of C4.5, boosting applied on C4.5, query by boosting applied on C4.5, and query by bagging applied on C4.5. For each data set, we performed 10-fold cross validation, with one-tenth of the available data (selected randomly) reserved as the test data and the rest used as the training data, or query data. For each of the 10

[2]This data set, abbreviated as the ICI ML repository in what follows, is available at URL address: "http://www.ics.uci.edu/~mlearn/MLRepository.html"

pairs of training and test data sets, we averaged the results over two randomized runs, a total of 20 runs for each data set.[3]

The query learning algorithms are used to pick the next query point from the training (query) data without replacement and are tested using the (separate) test data. When the specified number of candidates exceeded what is left of the training data, we went on with as many candidates as there were left. On one occasion, we also examined their predictive performance on the *query* data, from which query learners have selected a subset to learn from, instead of using the separate test data.

Finally, the parameters T and R in all the query learning methods were set at $T = 20$ and $R = 100$ in all of our experiments.

4 Experimental Results

We now discuss the results of our experiments on the UCI Machine Learning Repository. Figure 5 plots the learning curves obtained for the four learning methods on each of the eight data sets. Each graph plots the predictive accuracy (in percentage) of the four learning methods measured using the separate test data at every 50 trials. It is clearly seen from these graphs that in *all* eight data sets, the two proposed query learning methods achieve significant improvement in data efficiency as compared to C4.5. One can see that at very early stage in learning, say around 50 to 150 trials depending on the data set, the prediction accuracy of the query learning methods reaches a level that is achieved by C4.5 only towards the end of the data set. Table 2 gives concrete figures that quantify this observation. Here, 'the target error rate' was calculated using the error rate of C4.5 in the last 100 trials.[4] Then, we checked to see how many trials it took for all four methods to reach that error rate. In parentheses, we also exhibit the ratio of the number of trials required by each of the methods to that of C4.5. One can see that typically the data efficiency is improved by a factor of 2 to 4.

The speed-up achieved by the two query learning methods compared against boosting applied on C4.5 is less dramatic but still significant. From the graphs, one can see that on five of the eight data sets, namely breast-cancer-wisconsin, tic-tac-toe, ionosphere, house-votes-84 and wdbc, the advantage of the query methods over boosting is clear, while on the other three it is less obvious. These three data sets, crx, liver-disorders, and pima-indians-diabetes appear to have a common feature: That a certain level of accuracy is achieved with relatively few examples, but from then on the accuracy is hardly improved as the data size increases. It may be that the target function of these data sets is sufficiently noisy that no learning method can break this barrier. The increase in data efficiency achieved by the query learning methods in comparison to boosting is summaried in Table 3, similarly as before.

All the evaluation discussed thus far has been based on the prediction accuracy measured using *test* data, which are disjoint from the training data or the query data from which the query learning methods selected query points. As we remarked earlier, this is selective sampling and not genuine query learning. If we measure the prediction accuracy of query learning algorithms with respect to the *query* data, then this would translate to a genuine query learning scenario, except the function being learned is solely defined by the query data, only on those points that are in the data. We took this view point and examined the learning curves for the four methods with respect to this measure. Figure 6 plots these learning curves for the eight data sets as before. One can more clearly see the effect of query learning here – with respect to all but one data set (pima-indians-diabetes), the accuracy of the two query learning methods rise much faster than either C4.5 or boosting on C4.5., typically achieving an increase in data efficiency of fator 3 to 6.

On one of the eight data sets, tic-tac-toe, we ran the analogous experiments as above using a randomized version of WMP1 as the component learning algorithm. Figure 4 plots the prediction accuracy achieved by each of the five methods at the end of every 50 trials. Note that query by committee can now be applied because we use a randomized component algorithm. Here much of the tendency observed using C4.5 carries over. Notice, however, that here the two proposed methods, query by boosting and query by bagging, out-perform query by committee. Also, in this case query by boosting seems to do better than query by bagging, at least for a wide range of data sizes. The relative performance of the competing query learning methods appear to depend on the component learning algorithm (and the learning problem). Note further that boosting and the query methods applied on WMP1 achieve much higher accuracy than those applied on C4.5 on this particular problem. Interestingly, WMP1 itself does not have a higher accuracy than C4.5, but both boosting and query by boosting applied on WMP1 are significantly more effective than those applied on C4.5.. This observation suggests that on component algorithms and problems on which boosting is effective, query by boosting may do better than

[3]The results involving WMP1 were obtained by averaging over 10 runs, not 20 runs.

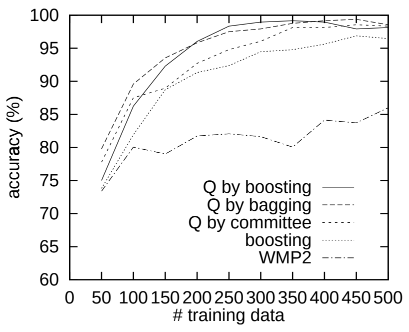
[4]For this calculation, we fed a randomly chosen test example after each trial, and the prediction error of the current trial was calculated by the average prediction error over the last 50 test trials.

name	query by bagging	query by boosting	boosting	C4.5	total size	target error rate (C4.5)
liver-disorders	86(0.30)	96(0.34)	108(0.38)	286(1.0)	310	0.3685
ionosphere	91(0.39)	97(0.41)	143(0.61)	236(1.0)	315	0.0935
house-votes-84	65(0.21)	72(0.24)	145(0.48)	303(1.0)	391	0.0465
wdbc	82(0.26)	88(0.28)	208(0.66)	314(1.0)	512	0.054
crx	64(0.50)	100(0.79)	119(0.94)	127(1.0)	621	0.171
breast-cancer-wisconsin	86(0.40)	83(0.39)	209(0.98)	213(1.0)	629	0.072
pima-indians-diabetes	67(0.44)	63(0.41)	81(0.53)	152(1.0)	691	0.2895
tic-tac-toe	236(0.39)	243(0.40)	308(0.51)	609(1.0)	862	0.1445

Table 2: Data efficiency increase achieved with respect to C4.5

name	query by bagging	query by boosting	boosting	C4.5	total size	target error rate (boosting)
liver-disorders	111(0.86)	126(0.98)	129(1.0)	-	310	0.3305
ionosphere	121(0.50)	119(0.49)	243(1.0)	-	315	0.073
house-votes-84	71(0.34)	136(0.65)	210(1.0)	366(1.74)	391	0.04
wdbc	97(0.32)	130(0.43)	300(1.0)	506(1.69)	512	0.0455
crx	86(0.60)	140(0.97)	144(1.0)	-	621	0.146
breast-cancer-wisconsin	103(0.34)	92(0.31)	301(1.0)	391(1.30)	629	0.0495
pima-indians-diabetes	99(0.56)	191(1.09)	176(1.0)	-	691	0.2475
tic-tac-toe	438(0.52)	517(0.62)	836(1.0)	-	862	0.053

Table 3: Data efficiency increase achieved with respect to boosting



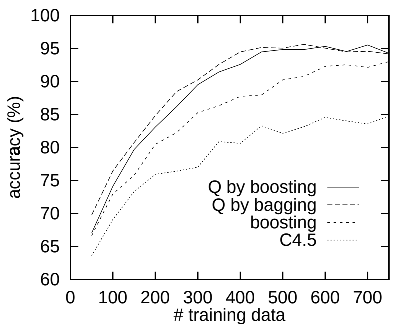


Figure 4: Prediction accuracy on test data on tic-tac-toe. Left: Using WMP1 as the component algorithm, Right: Using C4.5 as the component algorithm.

the other query learning methods as well.

The time complexity of all three query learning methods we considered is of the order $O(NTR \cdot F(N))$, where $F(N)$ is the time complexity of the component algorithm when run on an input sample of size N. This is a tractable but significant increase in computation cost as compared to the component algorithm. The judgement of whether the data efficiency brought about by these methods justifies the additional computational burden would depend on the exact application under consideration. Also note that both query by committee and query by bagging are parallelizable with respect to T and R, but query by boosting is parallelizable only with respect to R, and not T. Thus, only when query by boosting buys significantly more data efficiency, would it be the method of choice.

5 Concluding Remarks

We proposed two variants of query by committee that can be applied on an arbitrary component algorithm, be it deterministic or randomized, by incorporating the ideas of boosting and bagging. Experiments on data sets from the UCI Machine Learning repository demonstrated that, when using them with C4.5 as the component algorithm, the proposed query learning methods achieve significant increase in data efficiency as compared to both C4.5 and boosting applied on C4.5. On one of the data sets which can be cast as an n-ary learning problem, we tested these methods using a randomized weighted majority prediction algorithm for n-ary relations as the component algorithm, and found that the proposed methods performed better than query by committee. In the near future, we plan to carry out more systematic evaluation to verify the robustness of the proposed query methods on the choice of the component algorithm and the learning problem.

Acknowledgement

We thank Dr. S. Goto, Dr. S. Doi of NEC and Dr. K. Takada of NIS for their support. We would also like to thank those involved with the UCI Machine Learning repository for making the data sets available.

References

[ALN95] N. Abe, H. Li, and A. Nakamura. Online learning of binary lexical relations using two-dimensional weighted majority algorithms. In *Proc. 12th Int'l. Conference on Machine Learning*, July 1995.

[Ang87] D. Angluin. Learning regular sets from queries and counterexamples. *Inform. Comput.*, 75(2):87–106, November 1987.

[Bre94] L. Breiman. Bagging predictors. Technical Report 421, University of California at Berkeley, 1994.

[CS94] Mark W. Craven and Jude W. Shavlik. Using sampling and queries to exctract rules from trained neural networks. In *Machine Learning: Proceedings of the 11th International Conference*, pages 37–45, 1994.

[DSS92] H. Drucker, R. Schapire, and P. Simard. Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Information Processing Systems.* Morgan Kaufmann, 1992.

[Fre90] Y. Freund. Boosting a weak learning algorithm by majority. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 202–216. Morgan Kaufmann, San Mateo, CA, 1990.

[FS95] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory (EuroCOLT'95)*, pages 23–37, 1995.

[LC94] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning: Proceedings of the 11th International Conference*, pages 148–156, 1994.

[LG94] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. *Proceedings of 17th Annual International ACM-SIGIR Conference of Research and Development in Information Retrieval*, pages 3–12, 1994.

[PK95] G. Paass and J. Kindermann. Bayesian query construction for neural network models. In *Advances in nueral information processing systems 7*, pages 443–450, 1995.

[Sch90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[SFBL97] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, pages 322–330, 1997.

[SOS92] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 287–294. ACM Press, New York, NY, 1992.

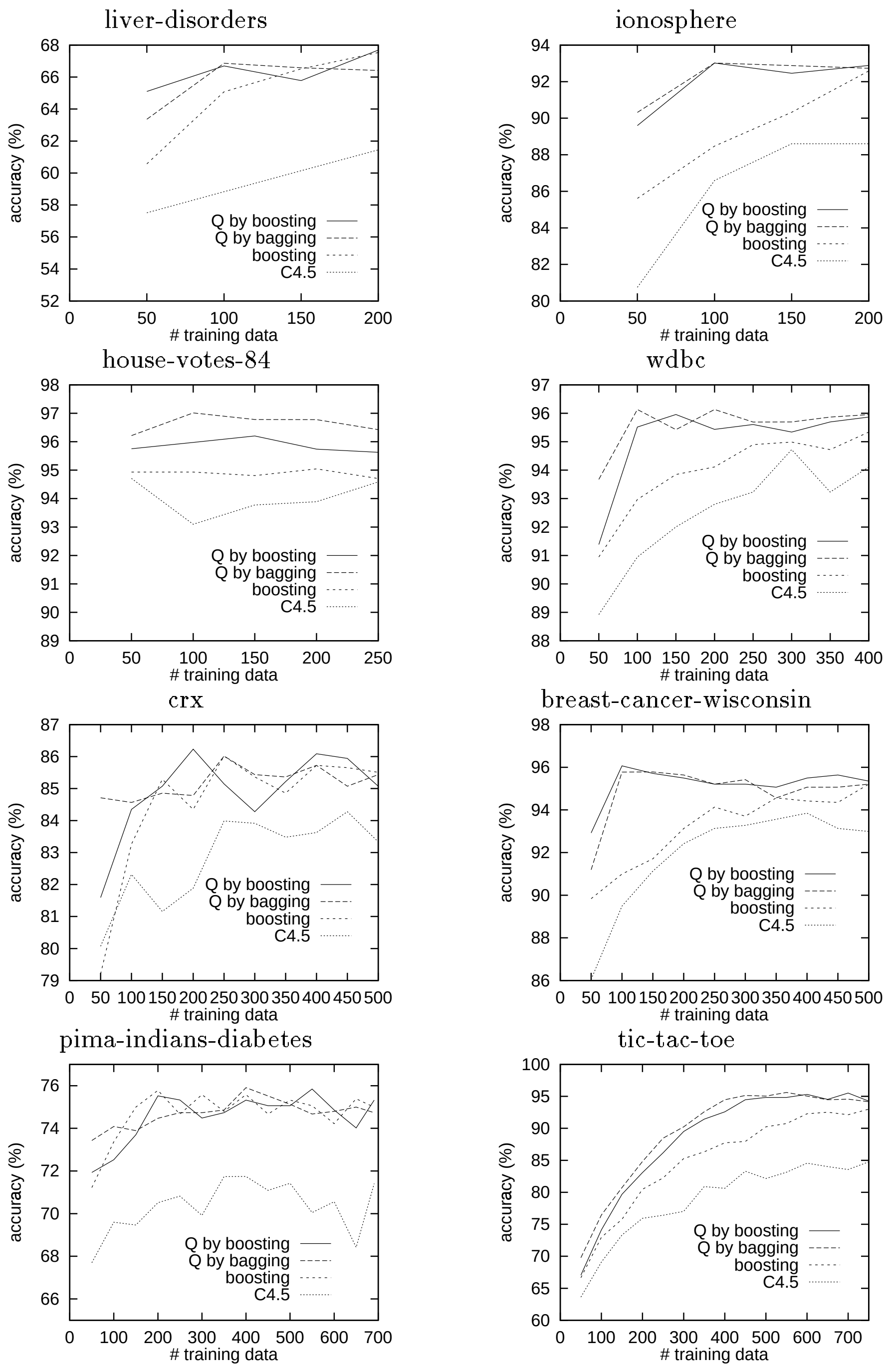


Figure 5: Learning curves for four learning methods on the UCI ML Repository.

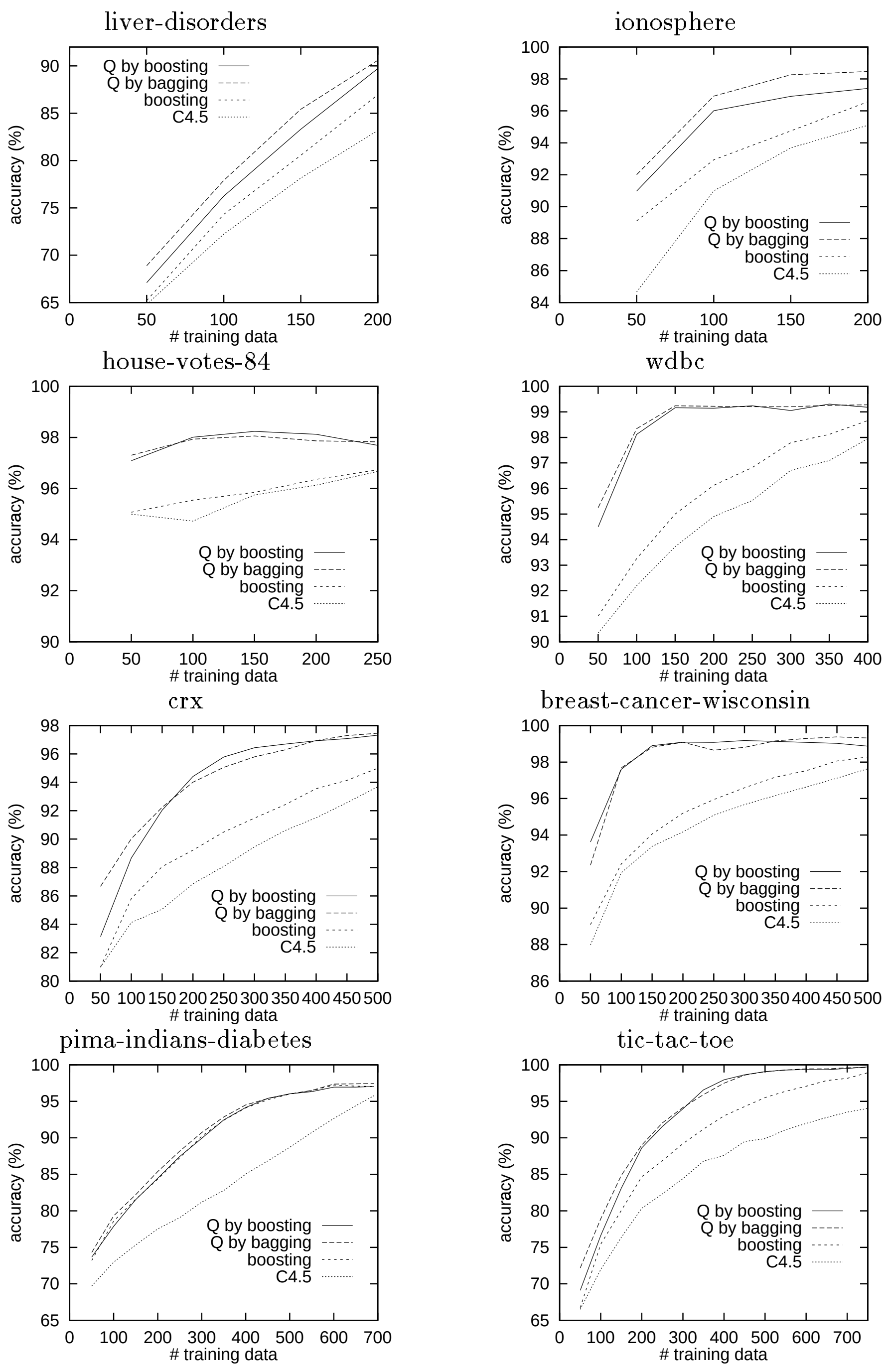


Figure 6: Learning curves on 'query data' for four learning methods on the UCI ML repository.